ISSN 0265-2919
80p 43

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication
IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

APPLICATION

TOPPING THE BILL We take a look at robots at the top end of the market 841

HARDWARE

OSBORNE ON SHOW The Osborne Encore is the latest lap-held from the company that started a revolution in microcomputing with the first true portable, the Osborne 1 849

SOFTWARE

VERTICAL TAKEOFF An introduction to a new series on vertical software — specialised applications written for specific groups of people like doctors and designers 844

SPORTING CHANCE Summer Games for Commodore 64 owners is an American game incorporating eight sports events 860

COMPUTER SCIENCE

REPEAT PERFORMANCE We show how to write some new control structures in LOGO and how to exploit the language's recursive abilities 853

JARGON

INPUT DEVICE TO INTEGER A weekly glossary of computing terms 852

PROGRAMMING PROJECTS

TAKE YOUR PICK We progress with the development of our adventure game by developing the routines that enable players to pick up and carry objects around 846

MACHINE CODE

OPERATING PRINCIPLES We begin a new series in which we explain in detail the different parts of the operating systems of all the popular micros 858

WORKSHOP

INSIDE JOB Our project to build our own floor robot continues as we connect the necessary plugs to the circuit board 856

REFERENCE CARD We list further extracts from the 6502 programmers' reference card INSIDE BACK COVER

Next Week

● The Compaq Plus was one of the first IBM compatibles on the market, we look at how it has stood the test of time.
● Concluding our series on robotics, we look to the future to see if machine will replace man.

Answers To Last Week's Quiz

1) The Touchmaster graphics pad has a socket marked 'foot switch' whose function is not explained in the documentation provided.
2) The Osborne Encore labels its function keys with icons. These represent pictorially the programmed functions, one of which is the built-in modem and communications facility.
3) The command interpreter routine in our BASIC adventure game will accept the half word 'LO' as a valid substitute for the complete word 'LOAF'.

DON'T VAT THE PRESS

There are strong reasons to believe the Chancellor of the Exchequer is planning to impose VAT on your magazine.
Such a move would turn the clock back 130 years — the last tax on newspapers and journals was repealed in 1855. Since then 'No tax on knowledge' has been a principle agreed by all Governments, even in the darkest days of war.
A free Press is a tax-free Press.
No Government should be given the power to impose financial pressure on a Press it may not like.
Tell your MP to say 'NO' to any tax on reading.

Issued by the Periodical Publishers Association, London

Editor Mike Wesley; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Art Editor** Claudia Zeff; **Chief Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Staff Writer** Stephen Malone; **Sub Editor** Steve Mann; **Consultant Editor** Steve Colwill; **Contributors** Geoff Bains, Harvey Mellar, Karl Dallas, Joe Pritchard, Steve Colwill, Chris Naylor; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Controller** Peter Taylor-Medhurst; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; © **APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA** – Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders.
UK/EIRE – Price: 80p/IR£1. Subscription: 6 months: £23.92. 1 Year: £47.84. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Price: 80p. Subscription: 6 months air: £37.96. Surface: £31.46. 1 year air: £75.92. Surface: £62.92. Binder: £5.00. Airmail: £5.00. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Price: 80p. Subscription: 6 months air: £43.94. Surface: £31.46. 1 year air: £87.88. Surface: £62.92. Binder: £5.00. Airmail: £8.30. **AMERICAS/ASIA/AFRICA** – Price: US/CAN$1.95/80p. Subscription: 6 months air: £51.74. Surface: £31.46. 1 year air: £103.48. Surface: £62.92. Binder: £5.00. Airmail: £9.45. **SOUTH AFRICA** – Price: SA R1.95. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Price: 80p. Subscription: 6 months air: £55.38. Surface: £31.46. 1 year air £110.76. Surface: £62.92. Binder: £5.00. Airmail: £9.85. **AUSTRALIA** – Price: Aus$1.95. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Price: NZ$2.25. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.
ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 6711. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.
NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.
ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited, Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH SHOT AT '192' KENSINGTON PARK RD, W.11

TOPPING THE BILL

In this part of our robotics series we turn our attention to the higher priced robots and robot arms available. Two distinct categories emerge: robots that are used as teaching aids to demonstrate the principles of robotics, and those that represent the state of the art of modern robot design.

Many of the robots that we discuss in this article are expensive — some cost over £1,000 — but they do not qualify as industrial robots and are designed for both home and school use. The first group that we will look at comprises robots that are engineered to a high standard and embody many of the features of industrial robot arms. The main difference between these arms and industrial arms is that most of the arms mentioned here are designed for educational use and are used to teach the principles of robotics.

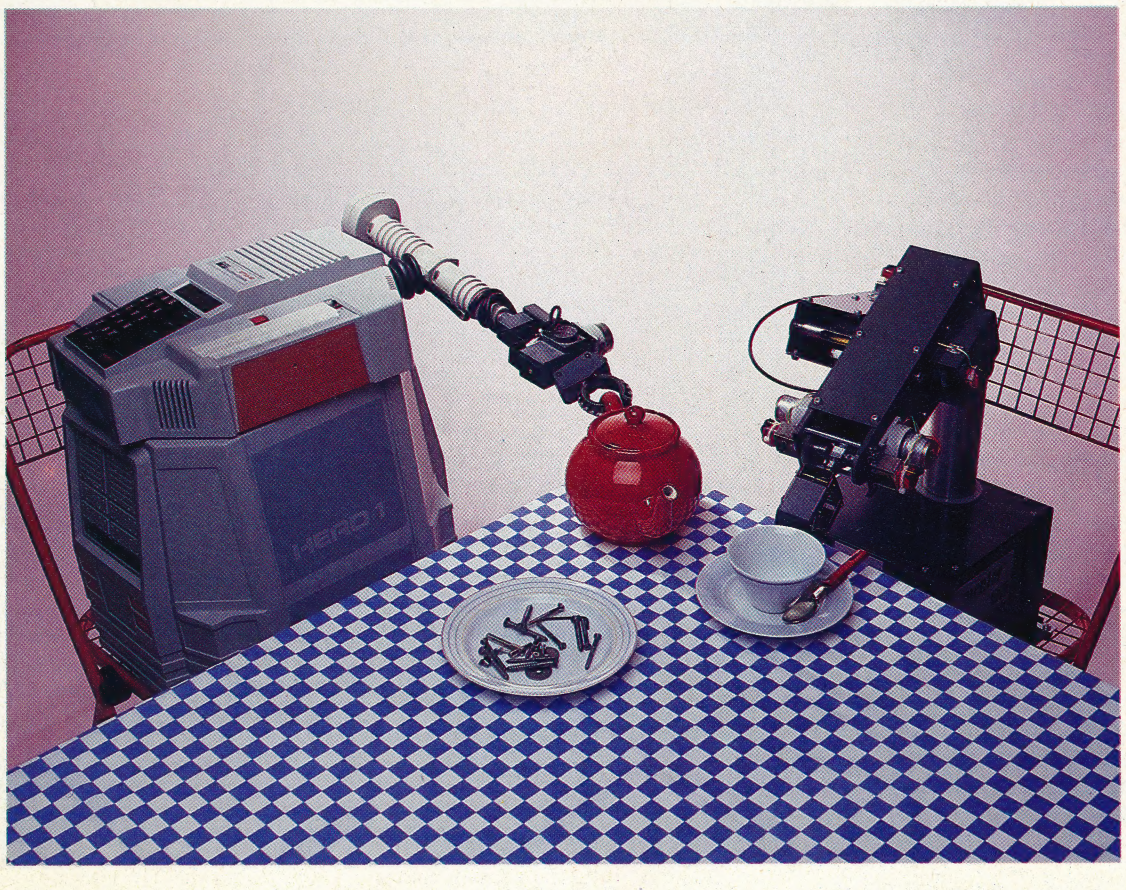
Typically, the main differences between these arms and their industrial counterparts is that these are smaller and less able to handle large objects. In many cases, of course, the educational market overlaps with the industrial market because if the industrial application requires only a relatively small, light arm then many of these arms would suffice. For instance, an industrial robot might well be needed to handle large steel ingots weighing several hundred kilograms. Equally, an industrial robot might be needed to assemble parts on a printed circuit board — a task that does not require a large and powerful arm. So, the robot arms in this category can be regarded as having the potential to carry out perfectly serious applications as well as being educational.

The second category that we shall be looking at concerns those robots whose design incorporates the very latest in contemporary robotic thinking. Many of them will be equipped with the sensory powers that we have discussed in previous articles in this series.

ARMS TO EDUCATE

A reasonably-priced robot arm is the Mentor from Cybernetic Applications. This sells for £345 in kit form, has six degrees of freedom (waist, shoulder, elbow and three axes of rotation at the wrist) and is powered by electric motors. It can be controlled from the BBC Micro, Commodore Vic-20 or the Sinclair Spectrum.

From the same company, but at the other end of



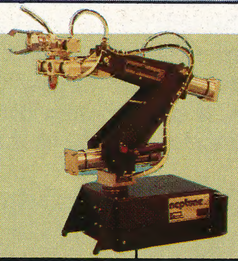
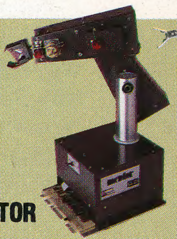
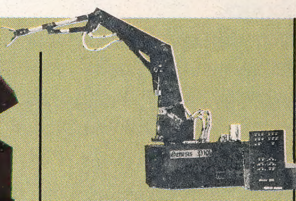
At Your Service
Because the Hero is both mobile and has a gripper, it can bring its owner a cup of tea in the morning (provided of course that the bedroom is not upstairs from the kitchen!) Our photograph shows the Hero and the Mentor enjoying a cup of tea

the price range, are the Neptune 1 and Neptune 2, costing £1,250 and £1,725 in kit form. These arms will lift up to 2.5kg and are hydraulically powered — although they use water and not the normal hydraulic fluid used in most robots of this type. These arms can also be controlled by the BBC

CHRIS STEVENS

Long Arm Of Learning

The two most significant areas of microelectronic research are chip design and robotics. Few hobbyists can experiment with new microchips in their home workshops, but robots are becoming more and more accessible. The primary purpose of the robot arms in the price ranges we have examined is to increase one's understanding of how robots work. The best way to increase our knowledge of robots is to examine working systems and try to improve them. The arms shown here, the Micro Grasp from Powertran Cybernetics and the Armdroid from Colne Robotics, are programmable through microcomputers and have five degrees of freedom. They can be purchased fully assembled or in kit form. The Micro Grasp costs £339.25 as a kit or £458.85 assembled. The Armdroid is £511.75 as a kit. or £569.25 assembled

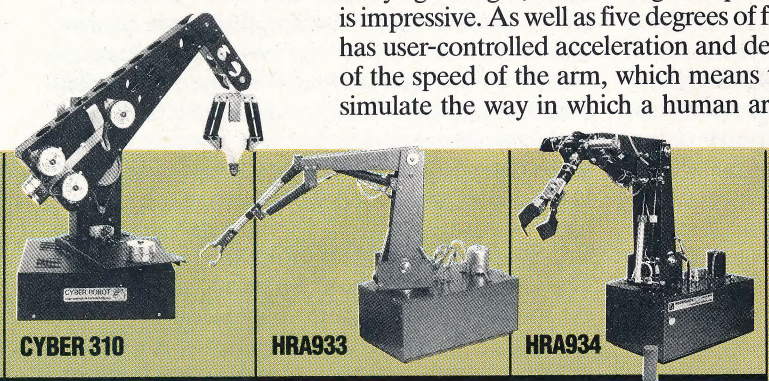
NAME	NEPTUNE 1	NEPTUNE 2	MENTOR	GENESIS P101	HERO
					
TYPE	Arm	Arm	Arm	Arm	Floor robot
PRIMARY PURPOSE	Educational	Educational	Educational	Educational	Educational/experimental
SENSORS	Potentiometer records position, gripper can sense how far it is closed	Potentiometer records position, gripper can sense how far it is closed	Potentiometer records position, gripper can sense how far it is closed	Positional feedback	Ultrasonic, allowing it to detect motion. Sensors detect 256 levels of light and sound. Tactile sensors on the gripper
DEGREES OF FREEDOM	6: waist, shoulder, elbow and wrist elevation, wrist roll, and gripper	7: waist, shoulder, elbow, wrist elevation, wrist roll, wrist yaw and gripper	6: waist, shoulder, elbow and wrist elevation, wrist roll and gripper	6: waist, shoulder, elbow and wrist elevation, wrist roll and gripper	4: shoulder, elbow, wrist and gripper
PRICE	£1,250	£1,725	£345	£1,750	£1,995
POWER	Hydraulic: water pump driven from mains	Hydraulic: water pump driven from mains	Mains	Hydraulic: water pump driven from mains	Rechargeable batteries
CONNECTS TO	BBC Micro, Spectrum, Vic-20	BBC Micro, Spectrum, Vic-20	BBC Micro, Spectrum, Vic-20	Programmed through a controller box; standard RS232 interface BBC Micro, Spectrum, Vic-20	Using a cross assembler, the Hero can be attached to any micro with a serial port
AVAILABLE FROM	Cybernetic Applications Ltd., West Portway Industrial Estate, Andover, Hampshire SP10 3NN	Cybernetic Applications Ltd.	Cybernetic Applications Ltd.	Powertran Cybernetics, West Portway Industrial Estate, Andover, Hampshire SP10 3NN	Zenith Data Systems, Bristol Road, Gloucester, GL2 6EE

Micro, Commodore Vic-20 and Sinclair Spectrum. The Neptune 2 has two different speeds of operation; this is useful as it can be made to move quickly when large arm movements are required and then slowed down for more precise work.

Powertran Cybernetics makes the Genesis P101, which has six degrees of freedom and sells for £750 in kit form. This model is hydraulically powered and comes with a controller box for programming the robot, together with a standard RS232 interface, which should enable it to be connected to most computers. This arm also comes preassembled and tested, but the price rises accordingly to £1,750.

An interesting mid-price arm is the Cyber 310 from Cyber Robotics. This is sold preassembled and costs £650. Versions of the robot are available for the BBC Micro, Jupiter Ace, Apple II, Commodore PET 3000/4000 and 8000 and the Hector HRX. It is powered by stepper motors and has a lifting capacity of only 250g, which makes it fairly lightweight, but the range of options it offers is impressive. As well as five degrees of freedom, it has user-controlled acceleration and deceleration of the speed of the arm, which means that it can simulate the way in which a human arm moves; altering its speed constantly as the nature of the task changes and simulating the effects of inertia. All the joints may be moved simultaneously and the position of the arm specified either as a relative position (for example, by instructing the arm to move forwards x units from its current position) or as an absolute position (by specifying a move to some point with reference to a 'home' position). It can be programmed in BASIC and also in a version of FORTH, known as Robo FORTH, developed by Cyber Robotics.

Moving right up in price we come to the HRA933 and HRA934 from Feedback Instruments. These cost £2,195 and £2,726 respectively and are sold ready-assembled. Both are hydraulically powered arms with five degrees of freedom and are capable of lifting 1.35kg with a positioning accuracy of 3mm. As well as position sensors for the arm joints, the arms also have tactile sensors in their end effectors. These sensors indicate when they have picked something up and enable the arm to control the force applied when picking up objects. Control is via an RS232 interface and specific instructions are given for control using the Apple II, the Tandy TRS-80, Commodore PET, AIM 65 and the MAT385.



CYBER 310	HRA933	HRA934
Arm	Arm	Arm
Educational	Educational	Educational
None	Ability to determine its position, gripper can sense how far it is closed	Ability to determine its position, gripper can sense how far it is closed
6: base rotation, shoulder rotation (which can bend through 180°) elbow, wrist elevation and wrist twist and grip	5:base rotation, elbow, wrist roll, wrist yaw and gripper	5:base rotation, elbow, wrist roll, wrist yaw and gripper
£650	£2,195	£2,726
Mains	Mains	Mains
BBC Micro, Jupiter Ace, Apple II, Commodore PET series and Hector HRX	Apple II, Tandy TRS80, Commodore PET series, AIM 65, BBC Micro, MAT 385	Apple II, Tandy TRS80, Commodore PET series, AIM 65, BBC Micro, MAT 385
Cyber Robotics, 61 Ditton Walk, Cambridge, CB5 8QD	Feedback Systems Ltd., Park Road, Crowborough, East Sussex, TN6 2QR	Feedback Systems Ltd.

STATE OF THE ART

The Hero-1 robot from Zenith Data Systems sells for £1,995 in kit form but offers some quite outstanding facilities. It is mobile and has an arm that makes use of a spherical co-ordinate system (see page 661) in which the arm is able to expand and contract telescopically. The Hero is equipped with a large array of sensors to detect movement, sound and light, including an ultrasonic distance sensor that helps it avoid collisions, and a speech synthesiser that gives the robot an unlimited vocabulary. It also has an arm with five degrees of freedom. The amount of assembly work involved in constructing the Hero 1 is considerable, so you may want it ready-made. In this case the price rises to a staggering £2,500.

All of these robots are, in some ways, little more than an expensive entertainment in terms of what they can actually do for you and, indeed, their main use to date has been to commercial firms who wish to use a robot for promotional purposes — handing out leaflets on exhibition stands, or product demonstration. However, they do represent the most up-to-date robotic technology available. They all use sensors in an intelligent fashion, move about intelligently and have intelligent arms. None of them has a vision system, but they can speak and they can hear acoustic signals and respond to them.

Needless to say, their price will prevent many people from buying them, but they do remain there as something to aspire to — something that you might perhaps be able to match by building a robot of your own. They also indicate the pace at which robotics is changing: a few years ago such robots would have been unthinkable — at any price.

VERTICAL TAKEOFF

'Vertical software' is programmers' jargon for specialised applications written for people like doctors, lawyers, designers, journalists, photographers and caterers. But often, generalised packages can be used in a special way, or can be customised for specific end-uses.

A great many uses exist for personal computers, and many of these applications are not readily apparent. They have developed as a result of creative adaptation of existing software, or the generation of special-use software. These are said to be 'vertical market' applications because they apply to a specific group of people, such as doctors, chemists or psychologists. In the series of articles we begin here, we will be looking at a number of vertical market packages that show off interesting new facets of microcomputer usage. The following examples will help to illustrate the kind of problems vertical market software is designed to solve.

A group of parents of teenage heroin addicts in London's West End is using Caxton Software's BrainStorm program for planning its campaign to spread awareness of their activities among local doctors, social workers and law enforcement agencies. A Kent restaurateur is using a Practicalc spreadsheet to analyse what customers are ordering, so that he can plan his future menus. A Sussex houseplant nursery is using the same program — working with four Commodore 64s — to handle everything from its large import project to controlling energy costs. A surgeon at a London hospital is using the Superbase program from Precision Software to help in his research into the causes and cure of cancer.

Chemists throughout Britain are using computers to comply with the new recommendation that all labels for patients' prescriptions must be printed, not handwritten. Kitchen designer Alan Batton of Warrington, Lancashire, uses a program — devised by a couple of snooker-playing friends to run on a BBC Micro — for shuffling stoves and fridges and other kitchen fitments around on a screen-based plan of the space available. In fact, it's been so effective in aiding his kitchen (and bedroom) fittings business, that he has gone into partnership with the program's original designers to offer the system to other retailers.

These are just a few examples of new answers to that time-honoured query from the prospective purchaser of a new computer: 'What can it do?' It's been estimated that the average computer user exploits no more than 10 per cent of the machine's capabilities — and that may be an over-estimate. Spending £200 or more on something that is then limited to one end-use, whether it be playing games or managing your accounts, is not as cost effective as exploiting its versatility to the full.

There are two ways of doing this. One option is to find new applications for standard software. Mike Ford, a professional photographer in Sheffield, uses the stock control module in the Anagram Integrated Accounts package to manage his library of photographs, which forces his file copies of prints and transparencies from past assignments to earn their keep. Similarly, some employment agencies use Tomorrow's Office database for matching the needs of their clients to the manpower available, keeping curricula vitae data on hard disk and mailing this out automatically. This is much cheaper than using packages designed for the job: it costs over a thousand pounds for AP Computer Consultants' Body Matching and Marketing package.

CUSTOM DESIGNED

The alternative is to seek a package designed for the special use you require. Imagine that you are a student of theology whose study desk is overburdened with massive tomes, concordances, commentaries, Bible dictionaries and the like. Well, then you may need 'The Word' Processor from Bible Research Systems for the IBM and compatibles. This package includes the entire King James translation with full search facilities for the creation of cross-references on disk.

And if you are suspicious of all translations of biblical material, you can check it out in the original with a program called The Greek Transliterator, which will give you the Greek equivalent of any English word or phrase, and display every occurrence in the English text. This allows you to compare the various ways in which a word has been translated. Such electronic Bible study is not cheap, however. Those two programs will set you back £253 each.

Let's consider another package, which at first would seem equally unlikely. If you're an engineer planning storm drains and sewers, then MIDUSS — the McMaster Interactive Design of Stormwater Systems — will help you to size pipes, channels and storage ponds, allow you to generate hydrographs, and give screen dumps of your high resolution graphics work. To achieve all this, you'll need a 256K Sirius, plus nearly £750 for the software.

Sewage systems seem to have provided programmers with a lot of stimulus. The rugged

Husky hand-held computer is at the heart of CAMIL — the Computer-Aided Manhole Inspection and Location program devised in Southampton and now being put into effect by water authorities all over Britain. The program allows surveyors to input data in the field and transmit it over telephone lines to host computers, which can then print out sewer layouts. The program can even respond to a request to see 'all brick sewers built before 1900' — and there's an awful lot of those.

Travelling salesmen will find extremely useful the 'Travelling' series of programs, which run on another hand-held — the excellent NEC PC-8201. In addition to basic packages like the Travelling Writer — a word processor for writing reports, with mailmerge and data management capabilities — the series includes the Travelling Project Manager, the Travelling Appointment Manager, the Travelling Sales Manager, and — this is most essential — the Travelling Expense Manager.

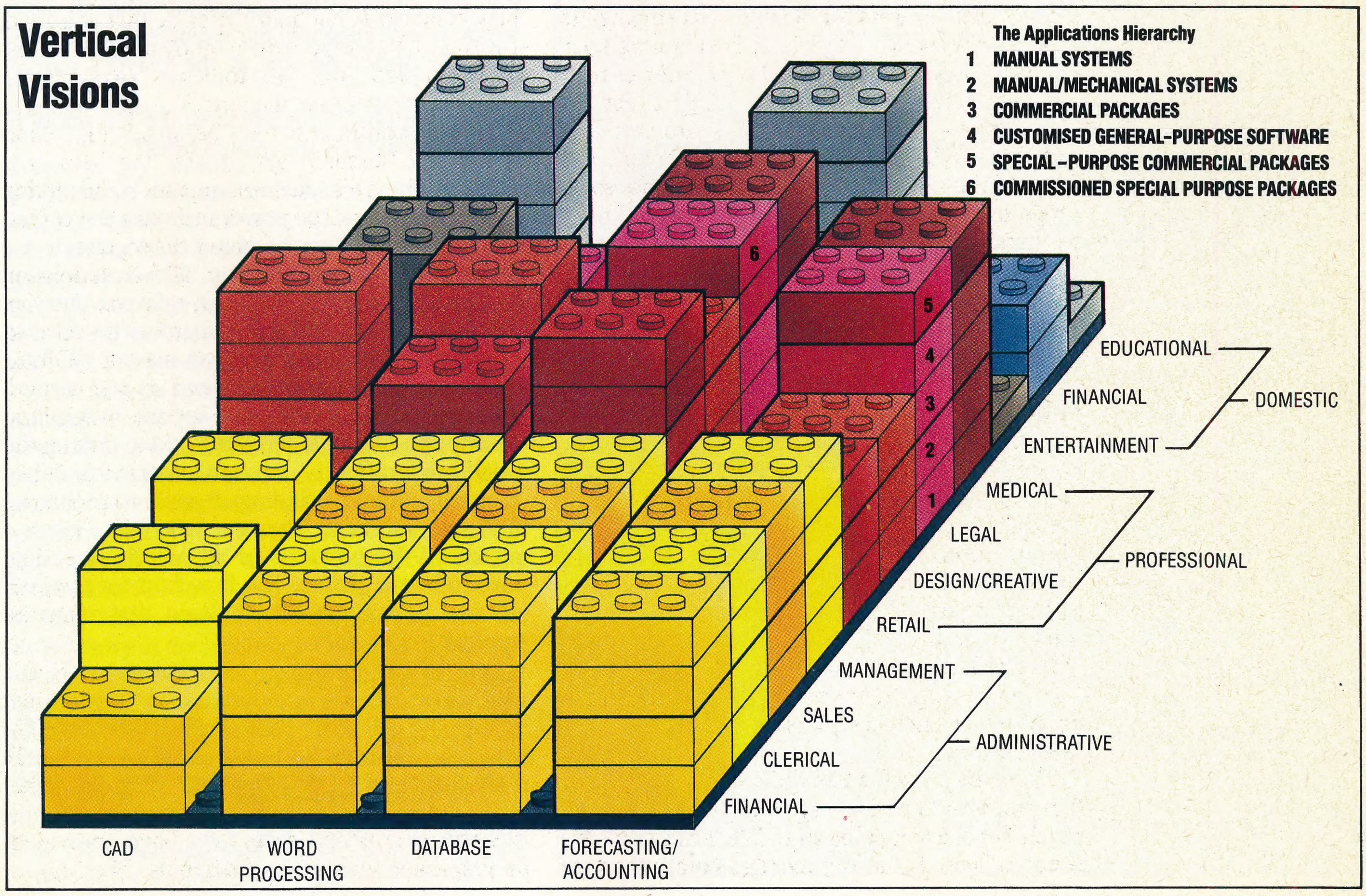
Salesmen, too, can improve their sales techniques with the Sales Edge module in the Human Edge suite of programs. These are distributed for the IBM and Apricot by Thorn EMI. The Sales Edge module ascertains the user's selling strengths and weaknesses with a series of agree/disagree questions and answers, and, after a similar set of questions about the client, evolves a suggested sales strategy with 'opening' and 'closing' gambits. Again, it's not cheap: £240. The complete set of Human Edge programs, including the Management Edge, the Negotiation Edge and the Communication Edge, costs £840.

If you're playing the money markets, then you may be tempted to invest £1132.75 in Forexia's Forextend, which allows you to study and analyse what's happening to the dollar, the pound, the Swiss franc, the Japanese yen and the Deutschmark. The program produces 37 charts of comparisons, relative strength indicators, interest rates and trade-weighted indices for every day within the time period between 1 October 1983 and the present day. A thousand pounds may seem a lot of money to many of us, but it's a lot less than you could lose in ill-informed currency speculations!

Scheduling and appointments programs are also becoming popular. Most computers have built-in clocks, but few of them are able to interrupt whatever you're doing to remind you where you should be. However, Hewlett-Packard's ROM-based schedule planner for their HP-75C can do just that — with a variety of different alarm sounds, as well.

In the next instalment of this series, we'll be looking in detail at BrainStorm, a program that claims to organise your thoughts in the way that word processors organise your sentences. And, subsequently, we'll be looking at software packages for people in sales, medicine, education, research, video, theatre and advertising, with specific user case histories.

Market Profiles
Software is written to meet the needs of the primary user groups, and to sell in the major market sectors. The bulk of it, therefore, comprises largely financial, word processing and database packages; a fourth area, computer-assisted design (CAD), is of continually growing significance. In our diagram we illustrate these application areas, and juxtapose them with the needs of a representative selection of computer users. The vertical axis represents the level of complexity of the application — manual methods lowest, off-the-peg computer software next and customised or purpose-built software at the highest level. The resulting use profiles show clearly that general-purpose software suits business needs very well in general but that specialist users need to customise existing packages or create their own — irregular high-low profiles show an unsatisfactory match between needs and resources, regular middle-height profiles indicate the market's ability to match needs and customers



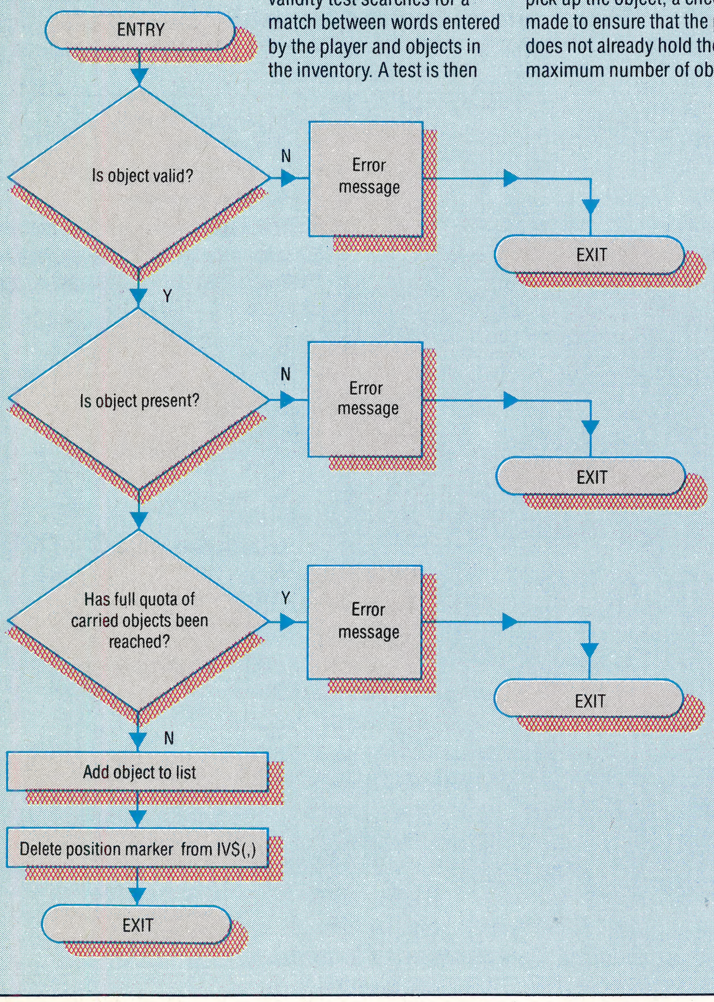
TAKE YOUR PICK

The first thing we did in this project was draw up a map of the imaginary adventure world of our game (see page 766). On this map, we positioned objects that will be of use to the player. Here we show you how to develop the routines needed to allow players to pick up and carry the objects around.

In the last part of the project we looked at command analysis and designated a group of 'normal' commands (see page 813). Included in this group were the commands TAKE and DROP, together with their variations PICK and PUT. Once the appropriate command has been recognised we can construct the routines that obey the command. We will first consider TAKE.

Objective Test

The flowchart for the TAKE routine shows tests used by the routine on the statement entered by the player. The validity test searches for a match between words entered by the player and objects in the inventory. A test is then made to ensure that the object in question is at the player's current location. Finally, before the player is allowed to pick up the object, a check is made to ensure that the player does not already hold the maximum number of objects



GARY CAPPS-JENNER

To understand the methods employed by the TAKE routine, let's recap on the way that the program keeps track of objects within the adventure world. In the first section of the project we designed DATA statements for each location that contained location descriptions, the names of objects present and information about the possible exits. After the data is read in, the array IV$(,) (used to store the object data for Haunted Forest) has the following contents:

N	**IV$(N,1)**	**IV$(N,2)**
1	GUN	10
2	LAMP	9
3	KEY	5

The first column of the array holds the object name, while the second contains its initial location number on the adventure world map. During the description of any location, the second column of this array is scanned to see if any of the objects are at the player's current location, P. When the player wishes to take an object from a location, using a command of the format TAKE THE OBJECT, several factors must be considered:

- Is the object in the command valid; in other words, does it appear in the inventory array, IV$(,)?
- Is the object present at the player's current location?
- Does the player already have the full quota of objects allowed by the game's rules?

If all of these considerations can be answered satisfactorily then the player may take the object. This involves adding the object description to the player's personal object array, IC$(), and deleting the position marker from the relevant entry in IV$(,). Note that the object name does not have to be deleted. If we use a position marker of −1 for each object that has been picked up and carried, then such objects will not appear in location descriptions.It would be rather odd to pick up the GUN from location 10, move to location 9 and then back to location 10, finding on your return that the GUN was still there. Thus the array IV$(,) keeps a record of the positions of all objects *not* being carried by the player. The flowchart for the TAKE routine shows the simple logic that must be applied.

```
3700 REM **** TAKE S/R ****
3710 GOSUB 5300:REM IS OBJECT VALID
3720 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:
     RETURN
3730 OV=F:GOSUB5450:REM CHECK INVENTORY
3740 IF HF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$
     (F,1):GOSUB5500:RETURN
3750 :
3755 REM ** IS OBJECT HERE ? **
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" IS
     NOT HERE":GOSUB5500:RETURN
```

```
3770 :
3780 REM ** ADD OBJECT TO LIST **
3790 A=0
3800 FOR J=1 TO 2
3810 IF IC$(J)="" THEN IC$(J)=IV$(F,1):AF=1:J=2
3820 NEXT J
3830 :
3840 REM ** FULL QUOTA **
3850 IF AF=0 THEN PRINT"YOU ALREADY HAVE TWO
     OBJECTS":RETURN
3860 :
3870 SN$="YOU TAKE THE "+IV$(F,1):GOSUB5500
3880 IV$(F,2)="-1":REM DELETE INVENTORY ENTRY
3890 RETURN
```

Let's now look at each of the three tests separately.

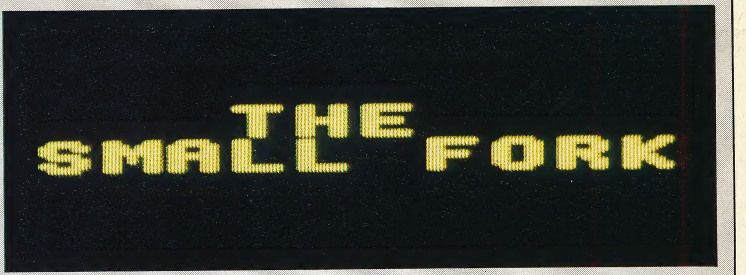
THE VALIDITY TEST

The most important and complicated of the three tests is the *validity test*. In its simplest form this could be a routine that just took the second part of the split command and compared it with each of the components of the inventory array, IV$(,). However, if this was the case, the TAKE command would be limited to the rigorous structure of TAKE OBJECT. Even variations such as TAKE THE GUN would be unacceptable as the routine would attempt to match THE GUN with the inventory, rather than just GUN. To allow flexibility in the command structure of TAKE we must develop a more sophisticated method of comparing the second part of the command given with the object inventory.

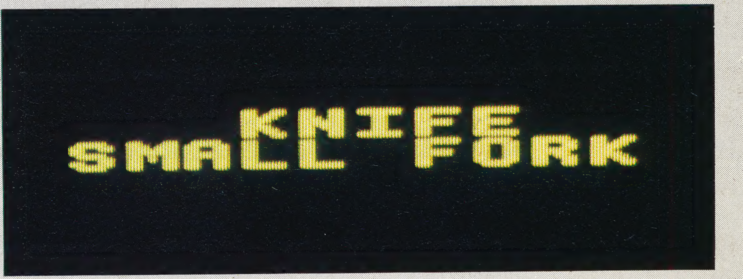
The most obvious method of increasing flexibility is to divide the second part of the command given into its constituent words and then compare each in turn with the object inventory. While overcoming the problem outlined earlier, this method too has its flaws. If, for example, we wanted to use a two-word description of an object such as LARGE KNIFE, then the command TAKE THE LARGE KNIFE would not give a match using this method. The routine would compare the words THE, LARGE and KNIFE separately with the inventory list. This problem can be overcome by making the routine even more sophisticated. Instead of searching for an exact match, a routine could be designed that would scan each object description in the inventory for the command word under examination, moving through the object name letter by letter until a match was found or the end of the object name was reached. The screen shots show how this is done.

One advantage of looking for a match between command and inventory in this way is that shortened versions of the object word can be given in the command. In the above example, the command TAKE THE KNI would also make the correct match, assuming that there were no other object names in the inventory prior to LARGE KNIFE with the combination of letters KNI. If this were the case, then an incorrect match would be made with the earlier entry in the inventory. Problems of this type are part of the price that must be paid for the greater flexibility of the routine. Most incorrect matching problems can be eliminated by careful selection of object names. If two object names must contain the same group of characters, or if one name is a substring of another — such as BULL and BULLET — then the shorter of the two names should be placed earlier in the inventory array. In addition, different

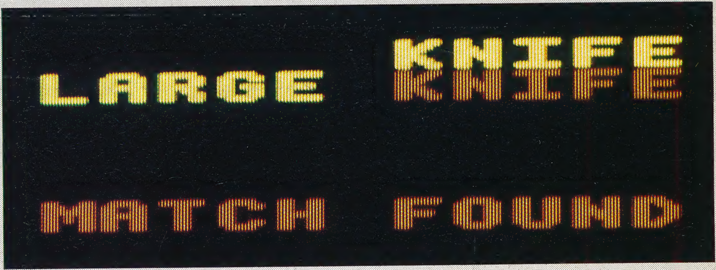
The Big Match



```
    THE
SMALL FORK
```



```
    KNIFE
SMALL FORK
```



```
      KNIFE
LARGE KNIFE

MATCH FOUND
```

```
  10REM **** MATCH AN OBJECT DEMO ****
  40MODE 5:COLOUR 2:DIM V$(3)
  60FOR I=1 TO 3:READ V$(I):NEXT I
  90A$="THE":B$="KNIFE":C$="KNI"
  95S$="                    "
 110M$=A$:GOSUB 1000:REM MATCH 'THE'
 130M$=B$:GOSUB 1000:REM MATCH 'KNIFE'
 150M$=C$:GOSUB 1000:REM MATCH 'KNI'
 160END
1000REM **** MATCH S/R ****
1010CLS:F=0:LW=LEN(M$)
1030FOR J=1 TO 3:LI=LEN(V$(J))
1042X=1:Y=6:GOSUB2000:PRINT S$
1045X=1:Y=6::GOSUB2000:PRINT V$(J)
1047FOR I=1 TO LI-LW+1
1050X=1:Y=5:GOSUB2000:PRINT S$
1060X=I:Y=5:GOSUB2000:PRINT M$
1070IF MID$(V$(J),I,LW)=M$ THEN F=I:I=LI:J=3
1075FOR D=1 TO 300:NEXT D,I:REM DELAY
1086IF F<>0 THEN GOSUB 2500
1087NEXT J:RETURN
2000REM **** POSITION CURSOR AT X,Y ****
2010PRINT TAB(X,Y);:RETURN
2500REM **** MATCH FOUND ****
2502COLOUR 1:X=F:Y=6:GOSUB2000:PRINT M$
2505FOR K=1 TO 5:X=1:Y=10:GOSUB2000:PRINT S$
2508FOR D=1 TO 500:NEXT:REM DELAY
2510X=1:Y=10:GOSUB2000:PRINT "MATCH FOUND"
2512FOR D=1 TO 500:NEXT D,K:REM DELAY
2520A$=GET$:COLOUR 2:RETURN
3000REM **** INVENTORY DATA ****
3005DATA"SMALL FORK","RED DOOR","LARGE KNIFE"

  10 REM ** SPECTRUM MATCH **
  40 INK 6:DIM V$(3,20)
1070 IF V$(J,I TO I+LW-1)=M$ THEN F=I:I=LI:J=3
2010 PRINT AT (Y,X);:RETURN
2502 INK 2:X=F:Y=6:GOSUB2000:PRINT S$
2520 A$=INKEY$:IF A$="" THEN 2520
2525 INK 6:RETURN

  10 REM ** CBM 64 MATCH **
  40 PRINT CHR$(158):DIM V$(3)
  50 DN$=CHR$(17): FOR K=1 TO 5:DN$=DN$+DN$:NEXT:
     DN$=CHR$(19)+DN$
2010 PRINT LEFT$(DN$,Y)TAB(X);:RETURN
2502 PRINTCHR$(28):X=F:Y=6:GOSUB2000:PRINT S$
2520 GETA$:IF A$="" THEN 2520
2525 PRINT CHR$(158):RETURN
```

The validity test subroutine designed for use with the TAKE routine scans the statement entered a word at a time, trying to find a match with an entry in the inventory. This short program illustrates the way in which the validity routine searches through the inventory for a match. Three objects are held in the inventory for this demonstration and the program tries to match the words 'THE', 'KNIFE' and 'KNI'. Whenever a match is found, the program waits for a keypress before continuing object descriptions that contain the same words, such as LARGE KNIFE should not be used.

```
5300 REM **** VALID OBJECT S/R ****
5310 NN$=NN$+" ":LN=LEN(NN$):C=1:F=0
5315 FOR K=1 TO LN
5320 IF MID$(NN$,K,1)<>" " THEN NEXT K:RETURN
5325 W$=MID$(NN$,C,K-C):C=K+1
5330 LW=LEN(W$)
5335 FOR J=1 TO 3
5340 LI=LEN(IV$(J,1)):REM LENGTH OF OBJECT
5350 FOR I=1 TO LI-LW+1
5360 IF MID$(IV$(J,1),I,LW)=W$ THEN F=J:I=LI:J=3:K
=LN
5370 NEXT I,J,K
5380 RETURN
```

Having found a match, the routine sets the variable F to the inventory array element that corresponds to the object in the command. If no match is found, then the value of F will remain zero, indicating that no such object exists in the game.

IS THE OBJECT PRESENT?

Once the array number of the object has been established by the validity test subroutine, the location of the object can easily be checked against the current location variable, P. The object to be picked up is in IV$(F,1) and its location is in IV$(F,2). Line 3760 of the TAKE routine in Haunted Forest checks this value against that of P. However, the error message generated — 'OBJECT is not here' — may not be strictly correct. The object may be present in the current location, but held by the player. Thus, a check should be made to see if the player is carrying the object in question before the error message is made. If so, a different error message (such as 'You already have the OBJECT') can be output. The following subroutine checks the main inventory and sets a flag, HF, to one if the object is being carried by the player. This condition is indicated by a -1 in the relevant element of the array.

```
5450 REM **** IS OBJECT HELD S/R ****
5460 HF=0
5470 IF IV$(OV,2)="-1" THEN HF=1
5480 RETURN
```

CHECKING THE PLAYER'S LIST

These two related tasks of checking to see if the player's list is full and adding to the list can be combined. Using the array IC$() to hold the objects carried, a FOR...NEXT loop can be used to locate the first free space in the array so that the new object may be entered. In the rules of Haunted Forest, a player may carry two objects only at any one time. Thus, the FOR...NEXT loop used is executed twice only. If a free space is found then the new object name is entered; if not then a message to the effect that the player is already carrying two objects is output to the screen.

The final task is to delete the position marker of the object just picked up from the inventory. This is simply done by setting IV$(F,2) to -1.

THE LIST COMMAND

Now that the player has the ability to pick up objects, we can include another command. It is often useful for the player to be able to see what objects are being carried. For example, if the player comes across a locked door he may have forgotten that he picked up a key 20 moves before. Allowing the player to list the objects carried serves as a useful memory jogger. The code required is simple: a FOR...NEXT loop is used to display the contents of the player's object inventory, IC$().

```
4100 REM **** LIST CARRIED INVENTORY ****
4110 PRINT"OBJECTS HELD:"
4120 FOR I=1 TO 2
4130 PRINT"    ";IC$(I)
4140 NEXT I
4150 RETURN
```

Digitaya Listings

```
2140 REM **** TAKE S/R ****
2145 IV$(4,1)="TICKET TO TRI-STATE"
2150 GOSUB5730:REM IS OBJECT VALID
2160 IF F=0 THEN PRINT"THERE IS NO ";W$:RETURN
2170 REM ** IS OBJECT ALREADY TAKEN ? ****
2180 OV=F:GOSUB5830
2190 IFHF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$(F
,1):GOSUB5880:RETURN
2200 :
2210 REM ** IS OBJECT HERE **
2220 IF VAL(IV$(F,2))<>P THENSN$=IV$(F,1)+" IS NOT
 HERE":GOSUB5880:RETURN
2230 :
2240 REM ** ADD OBJECT TO LIST **
2250 AF=0:FOR J=1TO4
2260 IFIC$(J)=""THENIC$(J)=IV$(F,1):AF=1:J=4
2270 NEXTJ
2280 :
2290 REM ** CHECK FOR FULL QUOTA **
2300 IF AF=0THENPRINT"YOU ALREADY HAVE 4 OBJECTS":
RETURN
2310 :
2320 SN$="YOU TAKE THE "+IV$(F,1):GOSUB5880
2330 IV$(F,2)="-1":REM DELETE POSITION ENTRY
2340 RETURN

5730 REM **** VALID OJECT S/R ****
5740 NN$=NN$+" ":LN=LEN(NN$):F=0:C=1
5745 FOR K=1 TO LN
5750 IF MID$(NN$,K,1)<>" " THEN NEXTK:RETURN
5755 W$=MID$(NN$,C,K-C):C=K+1:LW=LEN(W$)
5760 FORJ=1 TO 8
5770 LI=LEN(IV$(J,1)):REM LENGTH OBJECT
5780 FORI=1TO LI-LW+1
5790 IFMID$(IV$(J,1),I,LW)=W$THENF=J:I=LI:J=8:K=LN

5800 NEXT I,J,K
5810 RETURN
5820 :
5830 REM **** IS OBJECT HELD S/R ****
5840 HF=0
5850 IFIV$(OV,2)="-1"THEN HF=1
5860 RETURN

2540 REM **** LIST INVENTORY S/R ****
2550 PRINT"OBJECTS HELD:"
2560 FORI=1TO4
2570 PRINT"   ";IC$(I)
2580 NEXTI
2590 RETURN
```

Basic Flavours

Spectrum:

For the Haunted Forest listing make the following changes:

Replace SN$ by S$, IV$(,) by IC$() by I$() and NN$ V$(,), by R$.

5320 IF R$(K TO K)<>" " THEN NEXT K:RETURN
5325 LET W$=R$(C TO K–1)
5360 IF V$(I TO I+LW–1)=W$ THEN LET F=J: LET I=LI:LET J=3:LET K=LN

For the Digitaya listing, replace the same string variable names and make the same changes listed above, but for lines 5750, 5755 and 5790 respectively.

OSBORNE ON SHOW

The US-based Osborne Corporation, manufacturer of the first 'portable' microcomputer, the Osborne-1, has recovered from its financial problems to produce a new machine. Here we look at the Osborne Encore — the latest in a long line of IBM PC-compatibles.

As the first all-in-one portable computer, the Osborne-1 started a revolution in microcomputing. Equipped with a built-in monitor, twin disk drives and interfaces to modems and printers, the machine was the first self-contained CP/M business computer. Although the designation 'portable' was perhaps something of an overstatement (the Osborne-1 weighed 10.5kg), other manufacturers were quick to see the potential of the new machine and the Osborne Corporation, instead of being in a field of its own, found itself surrounded by competitors.

The crunch for the new company came in 1981 when IBM announced the launch of the first model in its Personal Computer range. The new machine quickly swept all before it as businessmen rushed to buy from the well-known giant of the computer industry. Osborne, together with many of its competitors, swiftly announced the imminent launch of a PC-compatible machine. This new model, the Osborne Executive, was to be equipped with dual processors, allowing it to run both CP/M and MS-DOS software. However, because of the worldwide shortage of 8086 microprocessors, the machine appeared without the chip needed to make it PC-compatible. As a result, sales of Osborne machines fell dramatically, and Osborne was forced to file for voluntary liquidation in the summer of 1983. But Osborne managed to survive and the new-look company is now a research and development operation similar to Sinclair Research in the UK.

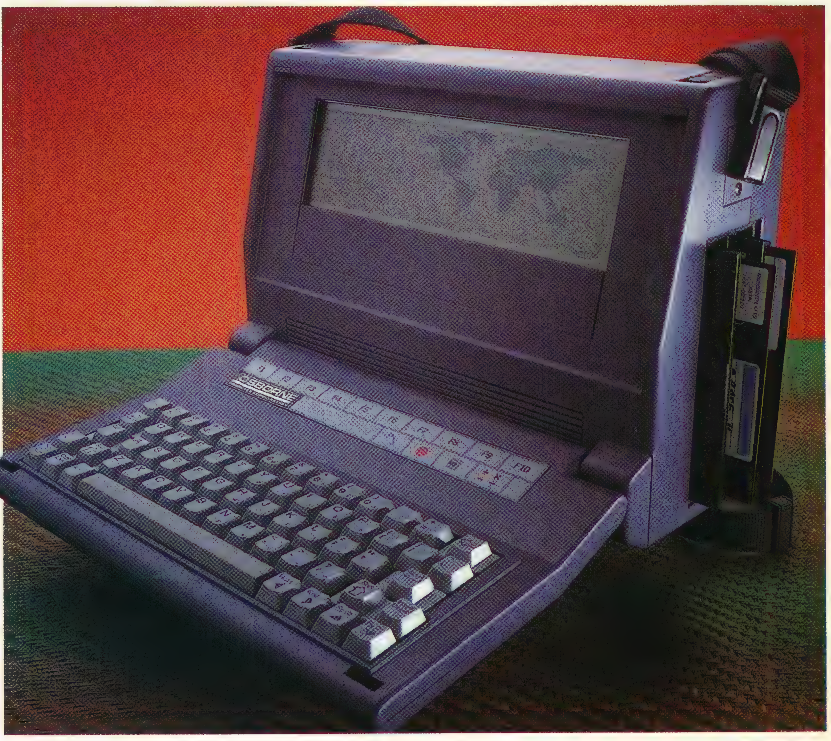
A new machine from the company — the Osborne Encore — has now finally arrived. It is a bold gamble from a firm that has barely managed to stay in business. Although not quite so compact as machines like the Epson PX-8, the Encore's significance lies in its attempts to bridge the gap between the lap-held and desk-top sections of the business market.

Weighing in at around six kilograms, the Encore is considerably less of a burden than its predecessor. The keyboard folds up into the screen, making a compact box that is roughly the size of three telephone directories. The casing is in sturdy blue plastic and the keyboard is held in place by a pair of clips.

The keyboard contains QWERTY typewriter keys, above which is a plastic membrane covering the function keys and the 'icons' (symbols that represent the built-in programs). On the main body of the computer itself is a 23.7cm by 8cm liquid crystal display screen.

The keyboard has a solid professional feel to it, with the control keys on either side of the keyboard and four cursor keys in the bottom right-hand corner. The design of the keyboard is very cramped, and this is, no doubt, a result of trying to fit all the features of the IBM PC into such a confined space. On the right-hand side of the IBM there is a separate numeric keypad that performs calculator functions. To maintain compatibility on the Encore, these keys have been incorporated onto the main body of the keyboard. The calculator functions are marked in blue, in contrast to the white labelling of the standard alphanumeric keys. The calculator is accessed, like the other built-in programs on the Encore, by pressing the appropriate icon on the panel above the keyboard. Encore icons represent the calculator, modem, disk and calendar routines.

The touch panel is not so well designed and detracts from the overall professional feel of the keyboard. The function keys (which on the PC are located separately to the left of the keyboard) have the same sort of feel to them as the ZX81 keyboard. Although one can feel the bubble



Compact Package
The Osborne Encore is one of the first IBM-compatible machines to be fitted with an LCD display instead of the standard cathode ray tube. When not in use, the keyboard folds up onto the screen, making a very compact package that can be carried with the shoulder strap provided

CHRIS STEVENS

membrane 'pop' underneath the panel, the keys lack the sureness of touch of a proper keyboard.

By incorporating an LCD screen, the designers have made the greatest savings in power consumption, since a liquid crystal display uses far less electricity than the usual cathode ray tube. It is here that the greatest problems with PC compatibility lie. It is not that an LCD screen lacks colours, as these can be easily substituted by varying the graphics shading; it is the size of the screen itself that is the major drawback.

The normal IBM screen has a text resolution of 80 by 25 characters but, because of development problems encountered by the screen's Japanese manufacturers in producing the equivalent LCD display, Osborne has been forced to introduce the Encore with a 80 by 16 display. This means that many packages written for the IBM will be unusable on the Encore. This is not a problem for programs that can be scrolled, but on packages where the display is 'paged' the user could run into serious difficulties, especially as prompts are generally shown at the bottom of a screen.

On the right-hand side of the computer, provision is made for a pair of 5¼in floppy disk drives, although the standard model will be equipped with a single drive only. On the opposite side is the power socket for the transformer, an on/off switch, a knob to adjust the contrast of the screen, and the battery housing, which can hold a special nickel cadmium battery pack to enable the machine to be run without access to a power point.

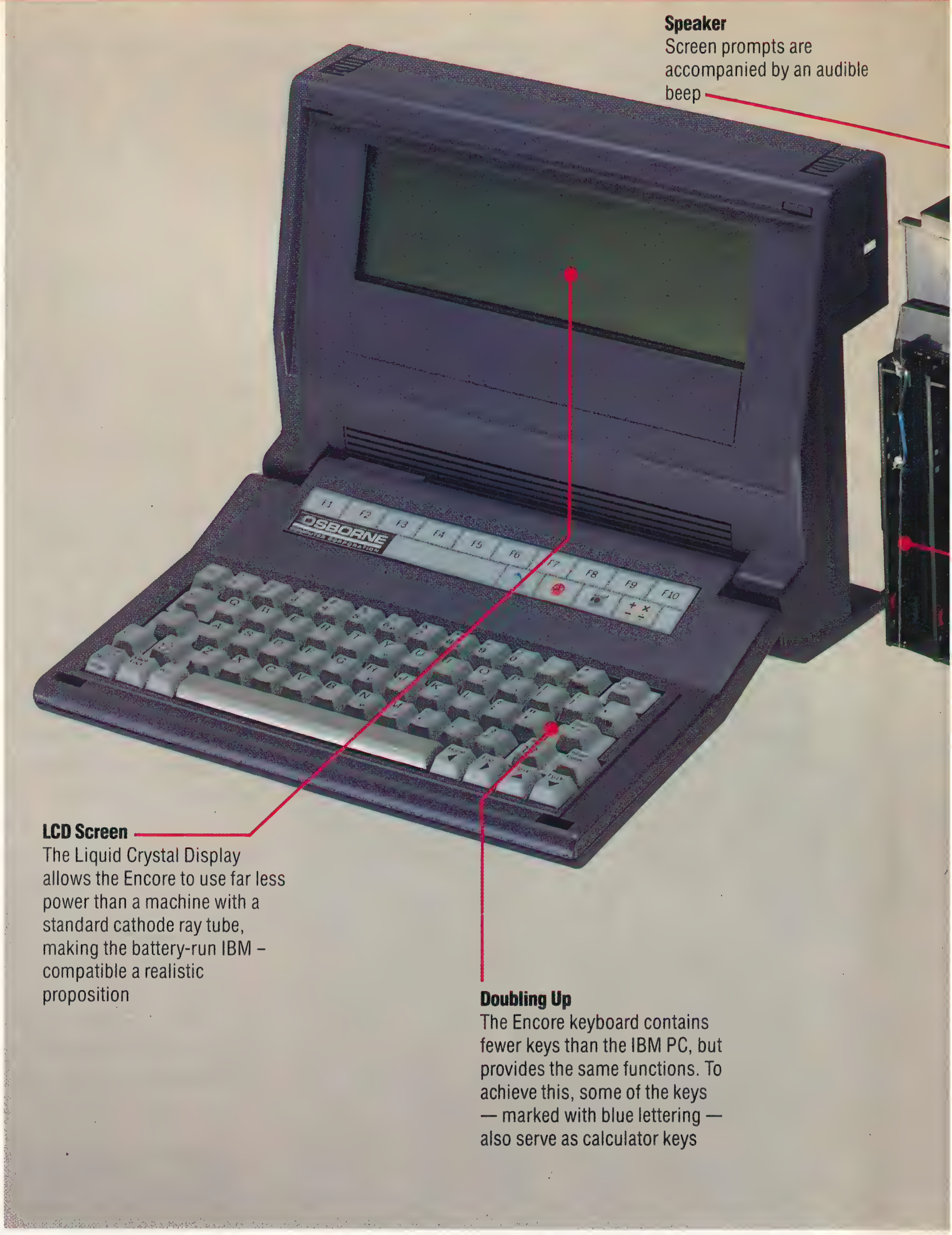
CONNECTING UP

At the rear of the Encore are the I/O ports. Reading from left to right, these comprise a telephone jack that connects to the Encore's built-in modem, a Centronics port to interface with a printer and an RS-232 serial port to connect with devices such as serial printers and modems.

To boot the MS-DOS system disk, you simply press the disk icon on the touch pad. The inbuilt programs can be run at any time, no matter what program is currently being used in the disk drive.

All IBM program disks that were loaded into the Encore were read by the computer. However, the screen restrictions made it difficult to detect whether the programs were running properly as many of the commands were entered 'blind'. Among the programs that the Encore successfully managed to execute was Lotus 1-2-3 — a notoriously difficult program for a compatible to run because of the unorthodox way in which the program accesses the IBM's built-in routines.

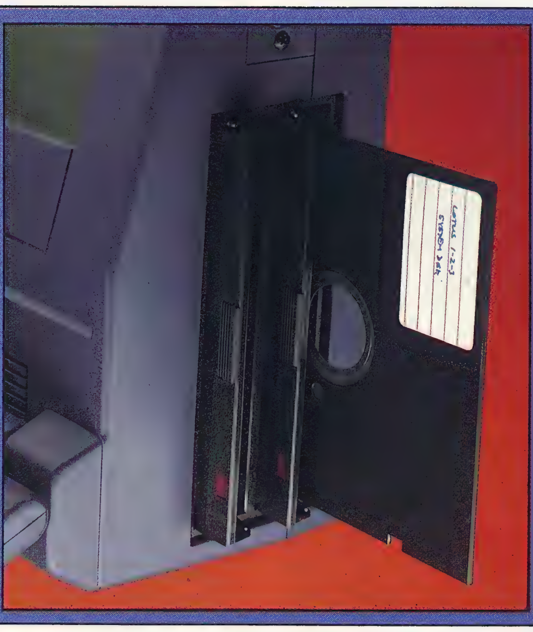
It is hard to tell if the Encore will prove to be as successful a machine as the Osborne-1. The screen takes some getting used to since, like all LCD displays, it needs strong light to produce sufficiently readable characters. This is not too much of a problem on lap-held machines with fairly large print, but the Encore's type is about half the size and it is doubtful whether many business customers will wish to take the time to get used to the display.



Speaker
Screen prompts are accompanied by an audible beep

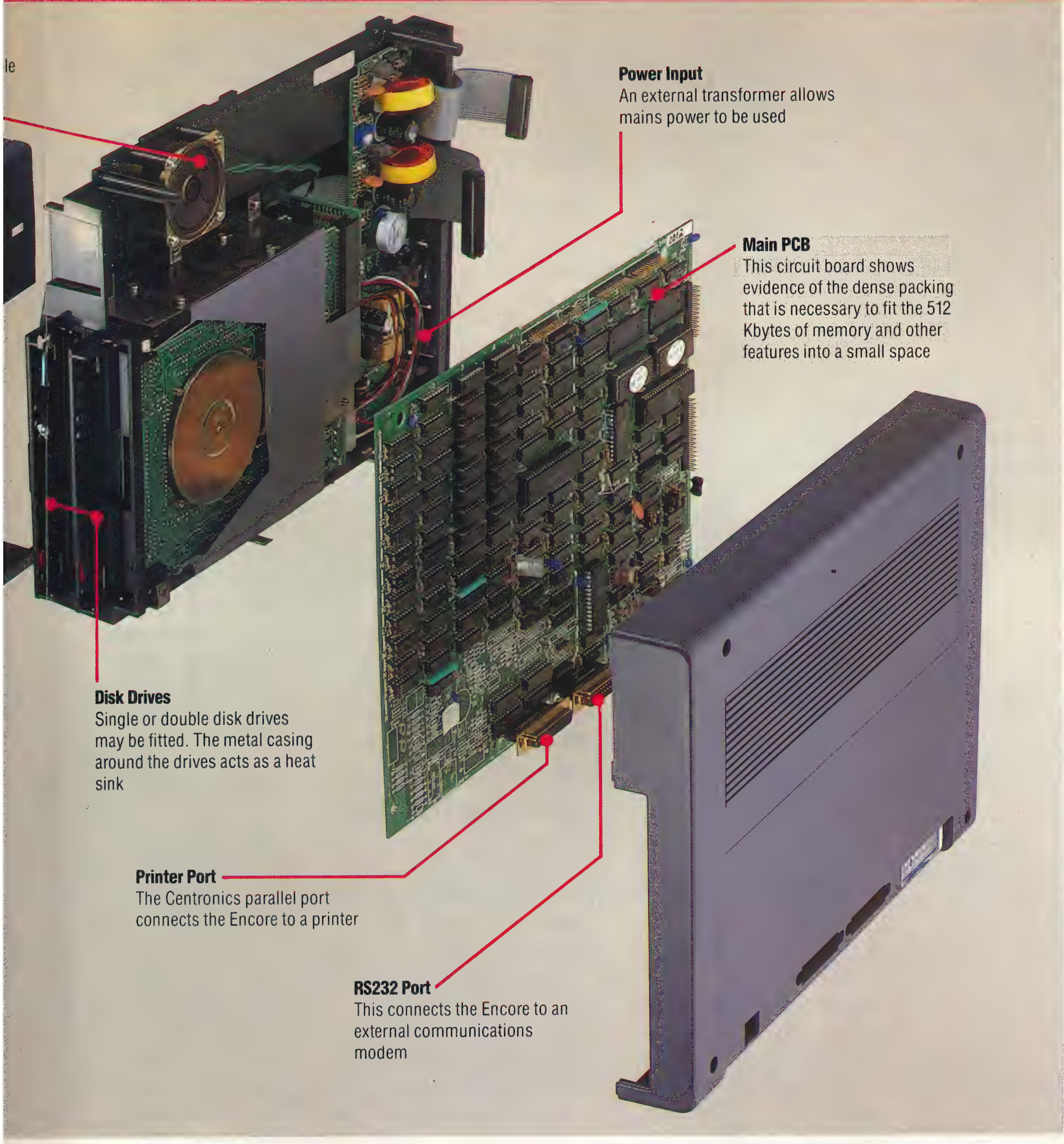
LCD Screen
The Liquid Crystal Display allows the Encore to use far less power than a machine with a standard cathode ray tube, making the battery-run IBM – compatible a realistic proposition

Doubling Up
The Encore keyboard contains fewer keys than the IBM PC, but provides the same functions. To achieve this, some of the keys — marked with blue lettering — also serve as calculator keys



Disk Disadvantage

The disk drives — either single or double drive units may be fitted — are located at the side of the machine. This makes the Encore a very compact package, but the user is forced to stretch to insert a disk or check whether the correct drive is being accessed



Power Input
An external transformer allows mains power to be used

Main PCB
This circuit board shows evidence of the dense packing that is necessary to fit the 512 Kbytes of memory and other features into a small space

Disk Drives
Single or double disk drives may be fitted. The metal casing around the drives acts as a heat sink

Printer Port
The Centronics parallel port connects the Encore to a printer

RS232 Port
This connects the Encore to an external communications modem

CHRIS STEVENS

OSBORNE ENCORE

PRICE
128K single disk drive £1,895; 128K double disk drive £2,195; 256K double disk drive £2,495; 512K double disk drive £3,095

DIMENSIONS
241×325×141mm

CPU
8086 16-bit microprocessor

MEMORY
This Encore is supplied with 512K of RAM

SCREEN
An LCD display of 80×16 characters or 480×128 pixels. Osborne has promised an 80×25 screen when it becomes available. Owners have been promised a cheap upgrade to the larger screen

INTERFACES
RS232 serial interface, RJ11 telephone jack and Centronics parallel port

LANGUAGES
Microsoft BASIC supplied on disk

KEYBOARD
62 typewriter-style keys and a keypad containing 10 function keys and four icon keys

DOCUMENTATION
The two user guides are, like all Osborne manuals, extremely well written, giving a full explanation of how to use the machine and containing plenty of examples

STRENGTHS
The Encore is an extremely powerful and compact machine that can be run on batteries for four to five hours

WEAKNESS
The screen is not yet fully IBM-compatible and can be difficult to read

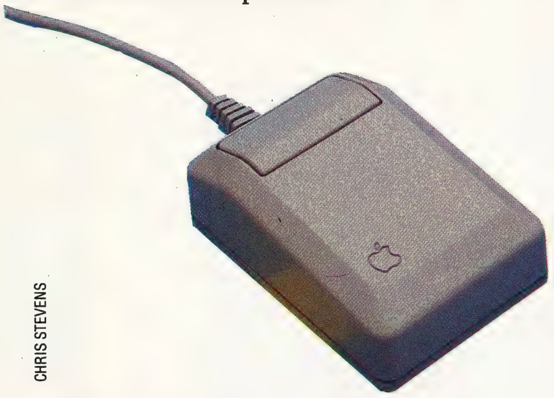
Time Check
The photo shows the 80×16 LCD screen. Note the grooves below the screen; these will allow the proposed 80×25 display to be fitted. Shown here is the calendar/diary display; this enables the operator to enter future appointments and set the time to any of the zones shown on the world map

Touch-Sensitive
Above the main keyboard is a touch pad — similar to the keyboard used on the Spectrum and ZX81 — that contains the 10 programmable function keys. Below the function keys are the icons that are used to call the Encore's ROM-based programs

I

INPUT DEVICE

An *input device* feeds information, in the form of digital electrical signals, into a computer or some other processor-based system. The most commonly-used input device is the keyboard, although other methods, such as speech input and the hand-held mouse, are becoming increasingly popular. Other input systems include light pens, graphics tablets and, of course, the joystick. Many mainframe computers use paper tape and punch card readers to input data.



Input Device
The mouse is an attempt to create an input device that is easier to use, particularly for novices to the standard typewriter keyboard

INPUT/OUTPUT

Input/Output (I/O) is the part of a computer system that handles the transfer of information to and from the central processing unit. Without a system of input/output, the CPU would be useless, since it would lack any communication with the outside world. The major purpose of input/output is to allow users to communicate with the processor.

The primary function of an I/O system, then, is to translate signals from the outside world into a form of data that can be understood by the CPU. However, this is only a part of the job of input/output. The system must also decide which of the several possible external devices (keyboard, joystick, printer, etc.) is being used at the time, and whether data is being written to or read from that device. Clearly, input/output tasks require a lot of attention — particularly if you consider external devices such as televisions, which have displays that need to be constantly updated. Therefore, most computers have a separate I/O chip set aside specifically to perform these tasks.

Input/output is most commonly associated with mass storage devices that can be used for either input or output, such as cassette tape decks and floppy disk drives.

INSTRUCTION

An *instruction* is a group of characters or digits that specifies an operation to be carried out by the computer. An instruction consists of an operation code (*op-code*), which is the action to be performed, and an *operand*, the data on which the operation is to be performed or its address. For example, in the instruction LDY 8, the op-code is LDY and the operand is 8.

Instructions can be grouped into three different types; *arithmetic* instructions, which perform arithmetical operations such as add, subtract, divide and multiply; *logic* instructions, which perform logical operations such as AND or OR; and *input/output* instructions, which perform I/O operations.

INSTRUCTION COUNTER

The *instruction counter* (also known as the 'program counter' or the 'current address register') is a register that is normally found in the CPU, which holds the memory address of the current instruction being executed. As the execution of a program proceeds, the instruction counter is added to in single steps, continually updating the address held as it moves through the sequence of instructions.

When the program encounters a branch or jump instruction, the instruction counter will have its contents altered by the operation. If this is a branch command, the current address held in the instruction counter will be PUSHed onto the stack, and the new address placed in the instruction counter. At the end of the subroutine, the address will be PULLed off the stack and placed back in the instruction counter. This will usually be done automatically by the processor and there is no need for the programmer to update the instruction counter manually.

INSTRUCTION SET

The *instruction set* is the list of instructions that a given microprocessor is able to perform. This is the irreducible set of instructions from which all other higher-level operations, such as BASIC commands, can be constructed. The instruction set is burned into the ROM on construction, and the extent and power of the instruction set can, therefore, vary according to the architecture of the chip itself. An instruction set should include operations to access and manipulate the contents of the registers and the accumulator. There should also be provision for branching — both conditional and non-conditional. Finally, there should be a number of instructions set aside to test and clear the status flags.

INTEGER

An *integer*, also known as a 'whole number', is a number that contains no fractional element. An integer can be positive, negative or zero. For example, 56, −43 and 0 are all integers, but 3.8, −2.001 and $3\frac{1}{2}$ are not.

An integer is not recognised as such by a computer, it is simply that we interpret it that way. For example, taking the eight-bit binary pattern 10000001, we can look on it in several different ways:

Binary Number	Interpretation	Decimal Equivalent
10000001	Unsigned integer	129
10000001	Sign and magnitude	−1
10000001	Two's complement	−127

REPEAT PERFORMANCE

This is the final instalment in the tutorial section of our series on LOGO programming. Here we show you how to add new control structures to the language, and explain how to write procedures that can themselves write procedures.

The LOGO primitive RUN takes a list as its input, and causes this list to be executed just as if it were a line of a procedure. This can be used to add new control structures to the language as and when they are required. So we could define a WHILE procedure as follows:

```
TO WHILE :CONDITION :ACTION
   IF NOT ( RUN : CONDITION ) THEN STOP
   RUN :ACTION
   WHILE :CONDITION :ACTION
END
```

Here's an example of how we could use it. POWER prints all the powers of its input below 1000:

```
TO POWER :X
   MAKE "P :X
   WHILE [:P < 1000] [PRINT :P MAKE "P :P * :X]
END
```

Control structures, such as WHILE, REPEAT and FOR, are common in other languages, but they are not really necessary in LOGO. A more natural way to write POWER in LOGO would be:

```
TO POWER :P
   IF NOT :P < 1000 THEN STOP
   PRINT :P
   POWER P * :P
END
```

REPEAT is provided in all versions of LOGO, but it is not strictly necessary, since you could define an equivalent word, REPT, in the following way:

```
TO REPT :NO :LIST
   IF :NO = 0 THEN STOP
   RUN :LIST
   REPT :NO - 1 :LIST
END
```

RUN is an extremely useful primitive for more advanced LOGO work. A program can assemble a list and then pass it to RUN to have it obeyed. We'll see an example of this shortly.

TAKING PROCEDURES APART

First of all we must define a procedure to draw a triangle in the usual way:

```
TO TRI
   FD 50 RT 120 FD 50
   RT 120 FD 50 RT 120
END
```

Now type PRINT TEXT "TRI. The result should be:

The text of the procedure is given as a list of lists, where each 'inner' list is one line of the procedure. To see why there is an empty list at the start, define this replacement for addition:

```
TO ADD :A :B
   PRINT :A + :B
END
```

Now PRINT TEXT "ADD will give:

```
[:A :B][PRINT :A + :B]
```

Clearly, the first list contains the inputs for the procedure. So TEXT enables us to get inside a procedure and find out what is there. DEFINE, on the other hand, does the opposite: it lets us define a procedure as a list of lists without having to go into the editor. Now try DEFINE "L [[:A] [FD :A] [RT 90] [FD :A / 2]] and then run L using, for example, L 30. Using DEFINE in immediate mode in this way has no advantages over using the editor. The advantage that DEFINE gives us is the ability of one procedure to create another procedure.

GROWING

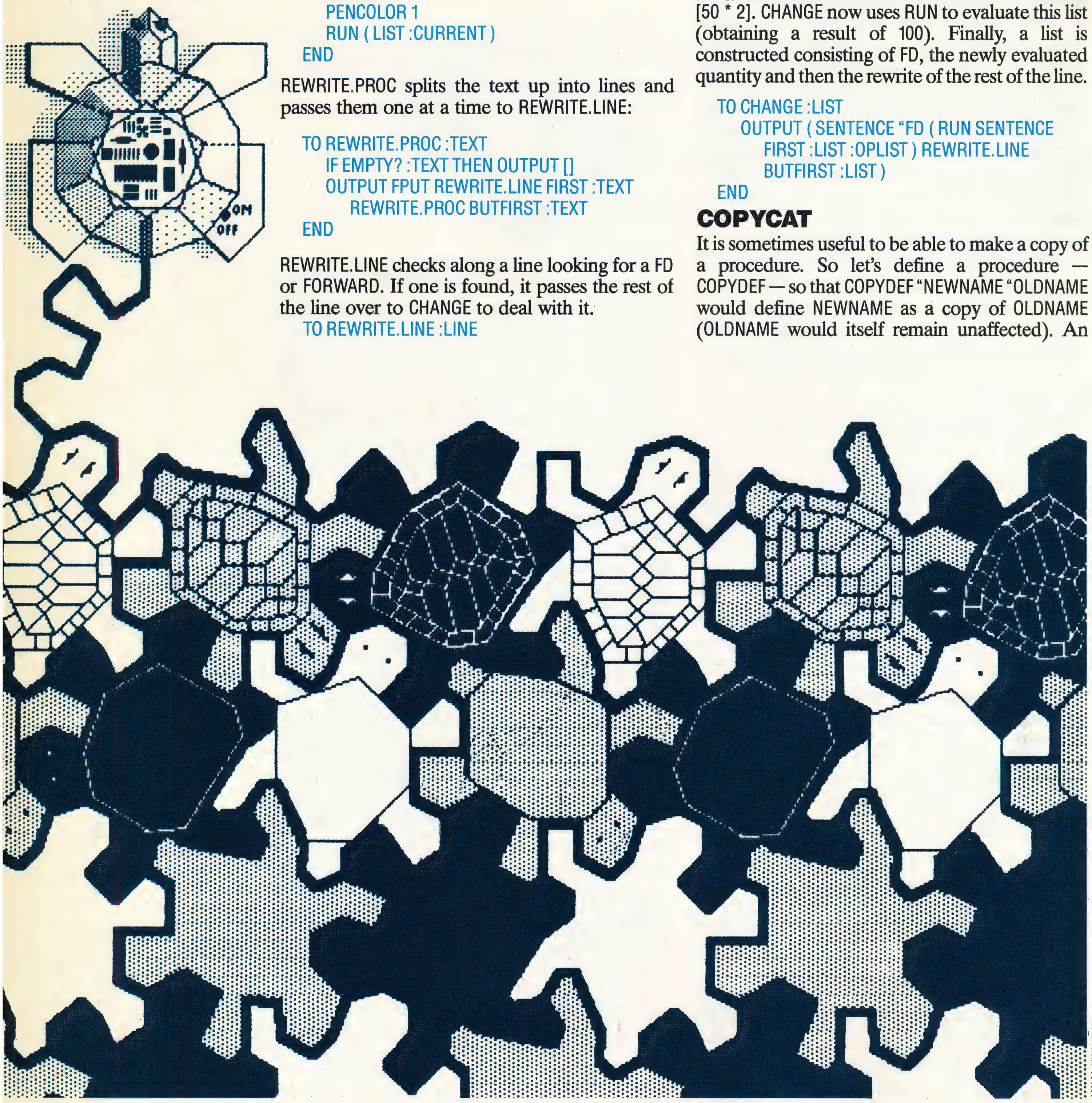
We are now going to develop a small system for investigating growth. The basic commands in our system are ASK, which selects the shape we will deal with, and GROW, which changes the size of the chosen shape. For example, ASK "SQUARE will draw a square, and then GROW [* 10] will erase the square and then redraw it with each of its sides increased by a factor of 10.

To keep the programs simple we will have to accept a few restrictions on what we can do with these commands. Firstly, the shape procedures given as inputs to ASK may not contain REPEAT or call subprocedures. Secondly, the system will break down if you get negative results. Neither of these problems is very difficult to deal with if you should wish to improve on what we give here.

ASK works by assigning the name of the shape to the global variable "CURRENT and then running the procedure. It does this by creating a list of one item — the procedure name — and then using RUN to execute it.

```
TO ASK :OBJECT
   HIDETURTLE
   MAKE "CURRENT :OBJECT
   RUN ( LIST :OBJECT )
END
```

Draw Me A Turtle
One cannot progress far in LOGO without coming across recursion, something defined in terms of itself. We have seen such examples as procedures that call themselves, lists defined in terms of lists, and now procedures to write procedures. With a little imagination, it would be easy in LOGO to create a drawing in the style of MC Escher, using the turtle to generate a turtle that draws a turtle...



GROW first wipes out the original drawing (Commodore LOGO uses PENCOLOR-1 for erasing), then uses DEFINE to define the current procedure as a rewritten one. The pen colour is then returned to normal and the new shape drawn. Note that the input to GROW is stored in the variable OPLIST — which we will need to use later.

```
TO GROW :OPLIST
   PENCOLOR -1
   RUN ( LIST :CURRENT )
   DEFINE :CURRENT REWRITE.PROC TEXT
      :CURRENT
   PENCOLOR 1
   RUN ( LIST :CURRENT )
END
```

REWRITE.PROC splits the text up into lines and passes them one at a time to REWRITE.LINE:

```
TO REWRITE.PROC :TEXT
   IF EMPTY? :TEXT THEN OUTPUT []
   OUTPUT FPUT REWRITE.LINE FIRST :TEXT
      REWRITE.PROC BUTFIRST :TEXT
END
```

REWRITE.LINE checks along a line looking for a FD or FORWARD. If one is found, it passes the rest of the line over to CHANGE to deal with it.

```
TO REWRITE.LINE :LINE
   IF EMPTY? :LINE THEN OUTPUT []
   IF ANYOF FIRST :LINE = "FD FIRST :LINE =
   "FORWARD THEN OUTPUT CHANGE BUTFIRST
      :LINE
   OUTPUT FPUT FIRST :LINE REWRITE.LINE
      BUTFIRST :LINE
END
```

CHANGE constructs the 'rewritten' line. The first item of LIST — the input to CHANGE — would have been the input to FORWARD in the original procedure. Say this is 50, and if OPLIST contains [* 2], then SENTENCE FIRST :LIST :OPLIST would be [50 * 2]. CHANGE now uses RUN to evaluate this list (obtaining a result of 100). Finally, a list is constructed consisting of FD, the newly evaluated quantity and then the rewrite of the rest of the line.

```
TO CHANGE :LIST
   OUTPUT ( SENTENCE "FD ( RUN SENTENCE
      FIRST :LIST :OPLIST ) REWRITE.LINE
      BUTFIRST :LIST )
END
```

COPYCAT

It is sometimes useful to be able to make a copy of a procedure. So let's define a procedure — COPYDEF — so that COPYDEF "NEWNAME "OLDNAME would define NEWNAME as a copy of OLDNAME (OLDNAME would itself remain unaffected). An obvious definition is:

```
TO COPYDEF :NEW :OLD
   DEFINE :NEW TEXT :OLD
END
```

The trouble with this definition is that if OLD does not exist then the procedure simply goes ahead and defines NEW as nothing. It would be better to pick up this problem and report on it. So a better definition of COPYDEF would be:

```
TO COPYDEF :NEW :OLD
   IF NOT PROCEDURE? :OLD THEN ( PRINT
     [THERE IS NO PROCEDURE] :OLD ) STOP
   DEFINE :NEW TEXT :OLD
END
```

This uses a procedure called PROCEDURE?, which outputs TRUE if its input is a procedure, and FALSE otherwise. PROCEDURE? and its counterpart, PRIMITIVE?, are very useful tests, but unfortunately they do not exist in MIT LOGO. So we've developed versions of PROCEDURE? and PRIMITIVE? that will work on both the Apple and the Commodore versions of LOGO:

```
TO PROCEDURE? :NAME
   IF NUMBER? :NAME THEN OUTPUT "FALSE
   IF LIST? :NAME THEN OUTPUT "FALSE
   TEST WORD? :NAME
   IFTRUE IF WORD? TEXT :NAME THEN OUTPUT
     "FALSE ELSE IF NOT (TEXT :NAME = [] ) THEN
     OUTPUT "TRUE
   OUTPUT "FALSE
END

TO PRIMITIVE? :NAME
   IF NUMBER? :NAME THEN OUTPUT "FALSE
   IF LIST? :NAME THEN OUTPUT "FALSE
   TEST WORD? :NAME
   IFTRUE IF WORD? TEXT :NAME THEN OUTPUT
     "TRUE ELSE OUTPUT "FALSE
END
```

AFTERWORD

We have now dealt with all the major features of standard LOGO, and have covered a wide area of possible applications. If you want to read more about the language, here are four suggested books:

- *Learning with Logo* by Daniel Watt (McGraw-Hill) is a wonderful introductory book, and is ideal for using with children.
- *Logo* by Harold Abelson (McGraw-Hill) is the 'standard' book on the language.
- *Turtle Geometry* by Harold Abelson and Andrea diSessa (MIT Press) takes a serious look at turtle geometry. The maths involved is at sixth form and college level — one of the later chapters develops a simulator for General Relativity in LOGO!
- *Thinking about [TLC] Logo* by John R. Allen, Ruth E. Davis and John F. Johnson (Holt Sanders International Editions) uses the rather idiosyncratic TLC LOGO, but the book is valuable for its investigation of artificial intelligence themes using LOGO.

Logo's Virtues

- It is interpreted, like BASIC, so it is easy to run and to modify programs.
- It is structured, with true procedures, unlike most versions of BASIC.
- It is extensible, like FORTH – i.e. it is possible to define new words that then become part of the computer's vocabulary.
- It has list processing, like LISP, so is useful for the exploration of areas such as artificial intelligence.

LOGO is not really designed for implementing fully worked out perfect algorithms, but is ideal for 'hacking'. We have written most of the programs in this series by starting with a simple procedure to perform just part of the task. We then alter the procedure in various ways, and as a result improve it and develop it as our understanding of the problem grows. At the end of the process we are left with a well-designed perfect algorithm.

Logo's Vices

- The main problems remain limited workspace and slow execution speed.
- Other features would be useful; in particular LOGO lacks error-trapping facilities, arrays and file handling. Some LOGOs do have these features but they are not common to all.

Logo Flavours

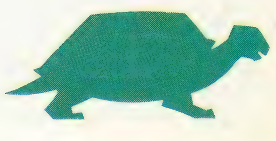
On all LCSI versions use:
NUMBERP for NUMBER?
LISTP for LIST?
WORDP for WORD?
EMPTYP for EMPTY?

For the different IF syntax, see page 815.

Spectrum LOGO has COPYDEF as a primitive, as well as PRIMITIVEP (corresponding to our PRIMITIVE?) and DEFINEDP (corresponding to our PROCEDURE?)

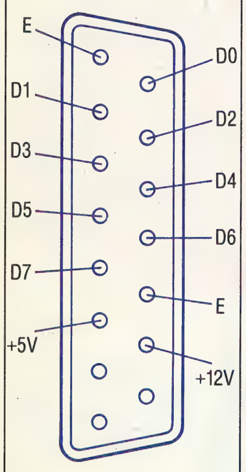
On the Atari use: PC for PENCOLOR, SE for SENTENCE, and HT for HIDETURTLE. DEFINE and TEXT do not exist in Atari LOGO, though the manual gives a way of defining them.

The pen colours used will differ from machine to machine. PC –1 is used here to erase lines, but some LOGO versions have this as a primitive PE (standing for 'Pen Erase')



INSIDE JOB

Now that we have mounted the motors and built the motor driver circuit board for our robot, we are in a position to connect the motors and D plugs mounted on the robot body to the circuit board and build a simple interface to the computer's user port. We shall also write a program to test the robot as constructed so far.



D SOCKET
VIEWED FROM REAR

Plug Connections

The user port plug diagrams show the connections for each type of plug. BBC Micro owners should use a 20-way ribbon cable and a snap-fit 20-way IDC socket. Noting the connections at the free end of the cable, the 11 wires required should be stripped, tinned, and matched to the connections on the interface board.

Commodore 64 owners should use a 24-way edge connector and a short length of 12-way ribbon cable. Note the connections to the plug and match these carefully to the connections on the interface board. As it is possible to insert this edge connector either way up it is important that the top of the plug be marked in some way.

Having made these connections we are now in a position to plug our interface cable into the robot and our user port. A 12v DC power supply should also be plugged in to the 2.1mm socket provided on the interface board

Bit Parts

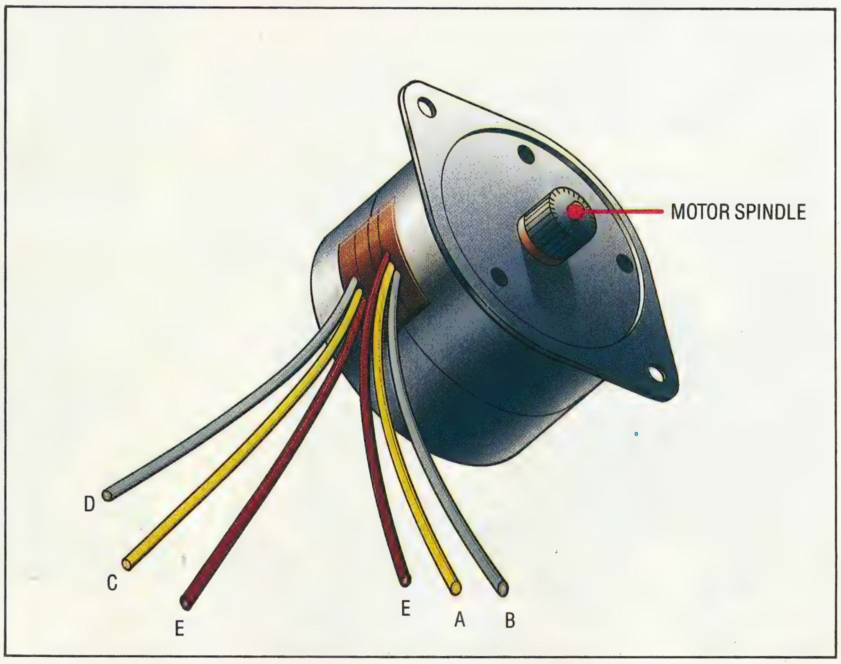
Now that the first phase of construction is complete we can write a short program to control the robot from the keyboard. Bits 0 to 3 of the user port data register control the motors. Bit 0 is the reset bit, normally set to 1; bits 1 and 2 control the right- and left-hand motor directions respectively. Bit 3 is the pulse bit that triggers the motors to turn through another step. The program uses the T, B, F and H keys to control direction and a repetitive loop to pulse the motors

```
1000 REM **** BBC ROBOT CONTROLLER ****
1010 DDR=&FE62:DATREG=&FE60:?DDR=15:REM LINES 0-3 OUTPUT
1020 PROCinitialise:REPEAT
1030 A$=INKEY$(1):IF A$<>"" THEN PROCtest_keyboard
1040 PROCpulse(10)
1050 UNTIL A$="X":?DATREG=0:END
1060 DEF PROCinitialise
1070 forwards=4:backwards=2:left=6:right=0
1080 dir=forwards:?DATREG=dir+1:ENDPROC
1100 DEF PROCpulse(m)
1110 FOR c=1 TO m
1120 ?DATREG=(?DATREG OR 8):PROCdelay(2)
1130 ?DATREG=(?DATREG AND 247):PROCdelay(2)
1140 NEXT c:ENDPROC
1150 DEF PROCdelay(n)
1160 FOR I=1 TO n:NEXT I
1170 ENDPROC
1180 DEF PROCtest_keyboard
1190 IF A$="T" THEN dir=forwards
1200 IF A$="B" THEN dir=backwards
1210 IF A$="F" THEN dir=left
1220 IF A$="H" THEN dir=right
1230 ?DATREG=((?DATREG AND 249)OR dir)
1240 ENDPROC
```

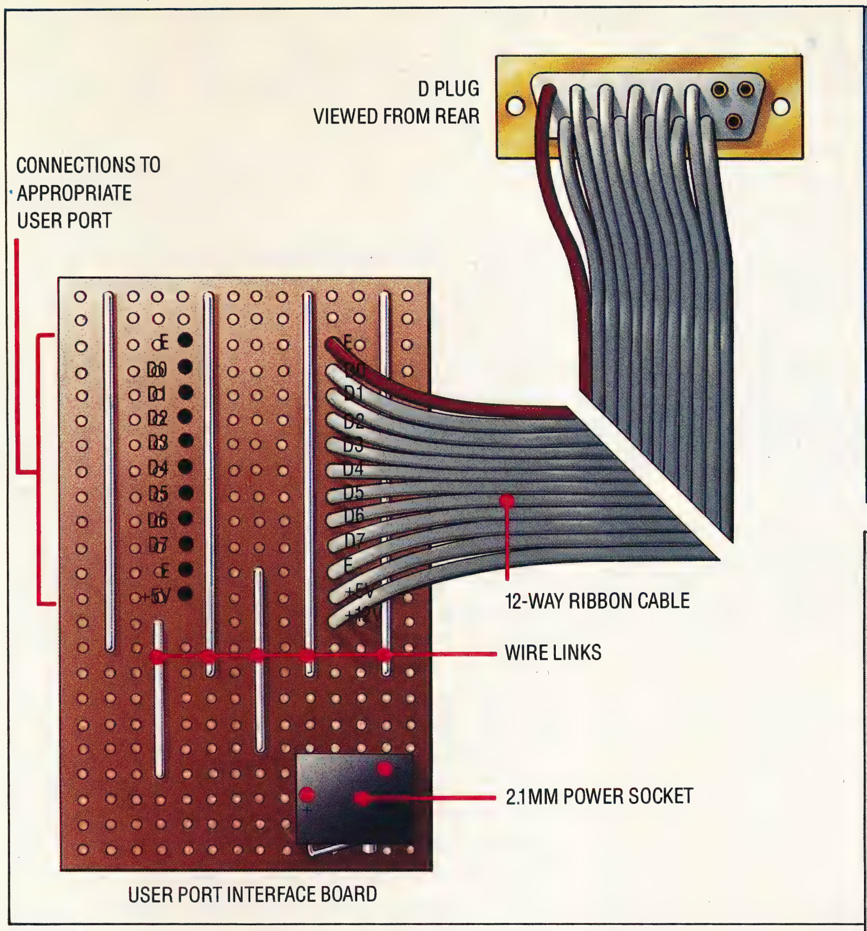
```
10   REM **** CBM 64 ROBOT CONTROLLER ****
20   DDR=56579:DATREG=56577:POKEDDR,15
30   GOSUB1000:REM INITIALISE
40   GETA$:IFA$<>"" THEN GOSUB3000:REM KEYS
50   M=10:GOSUB1500:REM PULSE
60    IFA$<>"X" THEN 40
70   POKEDATREG,0:END
1000 REM **** INITIALISE S/R ****
1010 FW=4:BW=2:LF=6:RT=0
1020 DR=FW:POKEDATREG,DR+1:RETURN
1500 REM **** PULSE S/R ****
1510 FOR C=1 TO M
1520 POKEDATREG,(PEEK(DATREG)OR8):GOSUB2000:REM DELAY
1530 POKEDATREG,(PEEK(DATREG)AND247):GOSUB2000:REM DELAY
1540 NEXT C:RETURN
2000 REM ****DELAY S/R ****
2010 FOR I=1 TO N:NEXT I:RETURN
3000 REM **** KEYBOARD TEST S/R ****
3010 IF A$="T" THEN DR=FW
3020 IF A$="B" THEN DR=BW
3030 IF A$="F" THEN DR=LF
3040 IF A$="H" THEN DR=RT
3050 POKEDATREG,((PEEK(DATREG)AND249)ORDR)
3060 RETURN
```

Missing Links

In order to connect the motors and D plug to the circuit board you will need to refer back to the illustration of the circuit board on page 837. This diagram shows where to solder the relevant wires to the board. Taking the motor connections first: each motor has six wires emerging from the motor housing. Take care to note that the wires emerge in two groups of three from the motor case. Each group of three wires has a yellow and grey pair and a single red wire. The yellow and grey pair labelled 'A' and 'B' emerge from the housing nearest the motor spindle as shown in the diagram. By relating these letters to the lettered connections for each motor on the circuit board diagram, solder each wire in place on the board



KEVIN JONES



Parts List

MAPLIN

No	Item	Source
1	15-way D socket	BK59P
1	15-way D cover	BK60Q
1	2.1 mm power socket	RK37S
1	20-way IDC connector (BBC)	FG87U
1	24-way edge connector (C64)	BK74R
1	strip self-adhesive pads	HB22Y

MISCELLANEOUS

4m	12-way ribbon cable
1m	20-way ribbon cable (BBC)
1	12v 1 amp DC supply

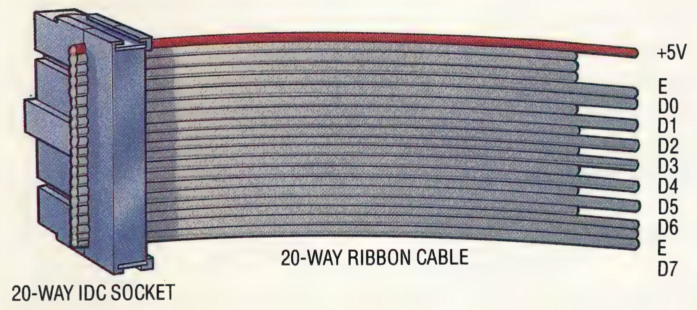
Interfacing To The User Port

Having completed the internal connections for the Workshop robot we have to design a simple interface board to allow us to control the robot from the user port and to supply the 12v DC required by the stepper motors.

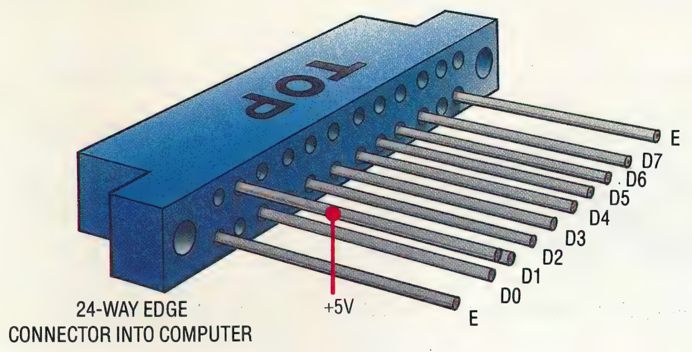
Cut a 24 strip by 14 hole piece of veroboard and connect three metres of 12-way ribbon cable to the board as shown. Using the red stripe down one edge of the ribbon cable as a guide, solder the 12 wires to the relevant pins of a D socket as shown and fit the D socket cover to secure the cable.

Mount the 2.1mm power socket on the board, noting the orientation of the pins. The central pole of this socket is negative. Then make the wire links as shown. The user port connections are also shown but here BBC Micro and Commodore 64 owners must part company as the user port plugs for each machine are different

BBC Micro



Commodore 64



Making Connections

Once the motor connections have been made it remains only to make the appropriate connections to the D plug, mounted on the lid of the robot body. The diagram shows the relevant pin connections for the D plug, viewed from the underside of the lid. Using a short length of 12-way ribbon cable, connect the data lines, D0 to D3, the +5v, the +12v and earth connections to the circuit board. Again, you must refer to the circuit board diagram given on page 837 for the correct wiring positions for these lines to the circuit board. Take special care to ensure that the +12v power line is connected to the correct point on the circuit board. Failure to do this could damage the internal circuits of your computer. Note that the data lines, D4 to D7, are *not* to be connected at this stage, as they are reserved for the input sensors we shall be adding to the robot in future instalments.

All the connections inside the robot are now complete. Find a suitable place inside the case for the circuit board to rest and secure it with self-adhesive pads. Close the lid and secure with the four corner screws supplied

OPERATING PRINCIPLES

The role of the operating system, or OS, is to serve as an interface between the programs written for a machine and its hardware. We begin a series of articles in which we examine the operating systems of several popular home computers — the BBC Micro, Sinclair Spectrum, Commodore 64 and others.

An operating system is written in the machine language used by the particular microprocessor incorporated in the computer. Thus, the BBC operating system is written in 6502 machine code and the Spectrum OS in Z80 code.

An OS consists of a range of routines that cover a variety of machine functions. For example, it will contain a routine that scans the keyboard for keypresses, something that the user will not need to bother about when writing software for the machine. In fact, in a good operating system, the user should be able to access any feature of the machine's hardware without needing to know the exact position in the computer's memory or input/output map of the routine that controls the piece of hardware in question. This means that hardware alterations to the machine are more easily made, and, as long as the OS has been altered to suit the new hardware, any programs written on older versions of a machine will run just as well on new versions of it.

The BBC Micro's operating system, for example, is very well-planned. A program written in BBC BASIC on a standard machine will run perfectly well on a BBC Micro equipped with a second processor — even though this is a radical hardware alteration to the machine. The BBC OS has gone through various stages of development and refinement to arrive at its current state of sophistication. Version 0.1, the first to appear, has now largely disappeared from circulation. It was made up of four EPROM chips, but lacked versatility and, in particular, could not support disk drives. Version 1.0, however, was able to support disks. It consisted of two eight-Kbyte chips on a small circuit board that fitted inside the machine. Version 1.2, the current version, is found in most BBC Micros in use today. A variant of this version — 1.2(US) — is designed for machines on the American market.

You can find out what version your machine contains by typing the command `*FX0` and then pressing Return. The version number of the OS will then be printed on the screen. The command `*HELP` will also return the OS version number, as well as listing to the screen the names of the ROM chips in the machine.

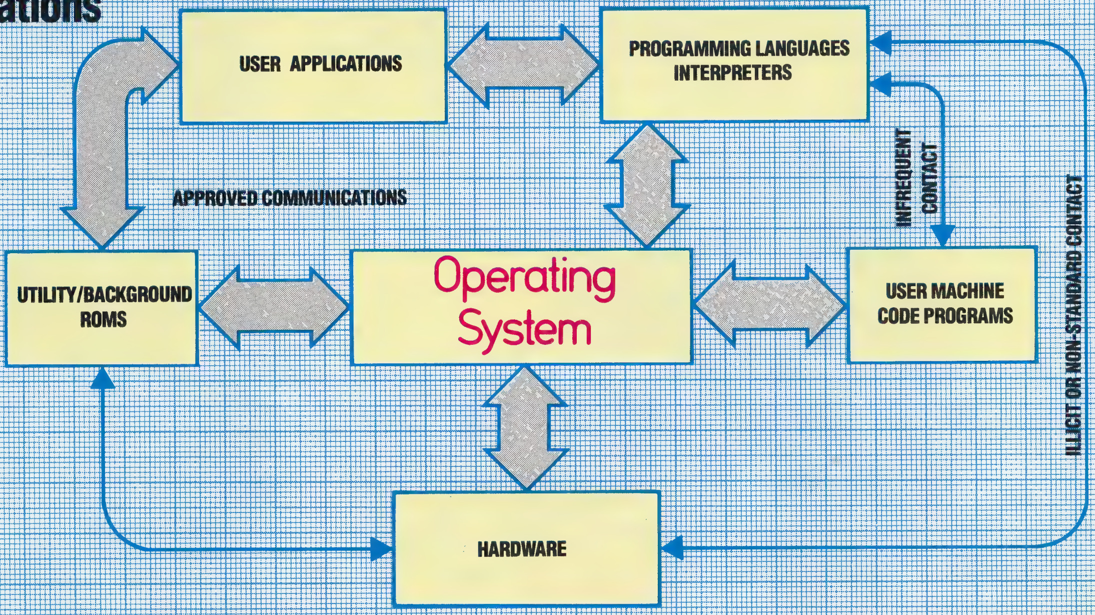
WHAT THE BBC OS DOES

The tasks the BBC operating system performs can be grouped into four main categories.

1. Input Routines: These routines accept information into the computer from what is called

Theatre Of Operations

The best operating system is the one you never see. To the user it should be a completely transparent interface between the application and the hardware, controlling the disk drives, spooling the output, driving the screen — making the system work. To the software and the hardware the OS should be a constant source of properly timed, correctly formatted input, and an ever-vigilant receptor of output in all forms. In return, system engineers expect that users and other engineers will avoid or discourage any system communications not routed through the OS



LIZ DIXON

the 'current input stream'. This is normally the keyboard, but other input streams are the RS423 interface and the current filing system — the latter is accessed via the use of the `*EXEC` command. The major OS routine that deals with input from all these sources is called OSRDCH, which stands for OS ReaD CHaracter.

2. Output And Display Routines: These routines handle output from the computer; in the BBC microcomputer there are a variety of output streams that can be selected as the current output stream, such as the television display, printer, RS423 interface and, using the `*SPOOL` command, the current filing system. As well as being responsible for printing or otherwise outputting data, the OS display routines are also involved with tasks such as the control of the 6845 display chip in the computer and the use of user defined characters, to name but two of the extra functions of these routines. The OS calls used are OSWRCH (for OS WRite CHaracter), OSASCII and OSNEWL.

3. Filing Systems: Any operating system must offer the user a means of saving the contents of computer memory to a more permanent medium. The section of the OS that deals with such transactions is called the 'currently selected filing system', and on the BBC we have the choice of tape, disk, Econet, ROM or telesoftware. The OS filing routines can make use of additional routines in ROM that instruct the OS to deal with the hardware associated with a particular filing system.

To enable simple interfacing of magnetic storage media via paged filing system ROMs, a number of standard OS routines are provided to deal with file-handling. These include routines for writing or reading whole files, getting or putting single bytes from an open file and reading or writing specified groups of bytes to or from a file. The seven operating system calls concerned with filing each use a vector to point to the relevant routine in the cassette filing system. If a paged filing system ROM is fitted, it can dovetail with existing BASIC or Assembly language file processing programs by simply changing these vectors to point to its own ROM routines. One ROM of this type is the DFS ROM that allows the BBC Micro to use floppy disk drives.

4. Interrupts: An interrupt is basically a signal generated in either software or hardware that tells the CPU to stop what it's currently doing and perform a task that requires immediate attention. After doing this, the CPU then resumes its task as if nothing had happened. In the BBC, there are a variety of interrupt-driven facilities that the user can gain access to via the OS.

In addition to these four main areas, there are two vitally important OS calls that deal with a lot of different functions of the machine. These are called OSBYTE and OSWORD, and are used to control the sound chip, the break key action, and similar things.

WHY USE OS CALLS?

In most cases, information about the internal arrangement of a computer's memory and hardware becomes available (legal injunctions permitting) within a few months of the release of the machine. If we know these details, why should we bother using operating system calls? Why not simply access the devices or memory directly?

Part of the answer to this question was mentioned at the beginning of this article: using OS calls offers you some protection from hardware or configuration changes made to the machine by the manufacturer. In a similar fashion, calling routines by using their addresses within the ROM will cause problems should the ROM be altered; if you access ROM routines through the appropriate OS calls then you will be protected from this problem.

The instructions below will, in a standard BBC Micro, write the value 200 to the BBC user port that is at address `&FE60`.

```
?&FE60=200
```

If a second processor is added to the computer, however, this routine will not send the value to the user port, but will write the value to a memory location in the second processor. By using the appropriate OS call we can get round this problem; the OS call will know how to write the value to the user port whether or not the second processor is connected. The OS call for this is:

```
*FX 151,96,200
```

This call is one of several that can be used to access areas of the BBC, such as the user versatile interface adaptor (VIA), the system VIA and the 1MHz bus.

Furthermore, why re-invent the wheel? If a routine to perform a particular function already exists within the machine, then why bother to go directly to the ROM routines? The chances are that the OS call will be more efficient than your direct access of the various routines involved in the OS call.

The only really good reasons for directly accessing memory or hardware devices are if you require speed of access or if an OS call doesn't exist to do the job. If speed is vital then direct access is often faster than going through the OS routines. However, it is still rather tricky to access the machine directly in this way, and we should take care in doing so — always remembering that programs that work on one particular machine may not work on other machines with different OS or hardware connected. This kind of problem is associated more with the BBC OS and the number of changes that it has undergone in its life time, than with the Spectrum, which has had the same OS throughout its history. In the next instalment of this series, we will begin to look at the BBC OS routines in more detail. Our discussion of the Spectrum OS, and other machines, will follow later in the course.

SPORTING CHANCE

The 1984 Olympic Games in Los Angeles generated a lot of interest in software based on competitive sports. Although some of these packages have originated in the UK (Daley Thompson's Decathlon is a recent chart-topper), most of them — such as the package we look at here, called Summer Games — come from the US.

Summer Games from Epyx is marketed in the United Kingdom by Quicksilva. The package includes eight sports events — from athletics and swimming to gymnastics and shooting. Each event can have up to eight different competitors, who can choose one of 17 countries to represent. Nationals whose flags are not included in the program have the opportunity to compete under the Epyx standard.

Once the game has loaded, the first scene shows the opening ceremony. To the accompaniment of suitably heraldic music, an athlete carrying the Olympic torch runs up to a rostrum and lights the flame, as a flock of doves is released into the sky. The sound and graphics of the opening ceremony are typical of those of the package as a whole. The background is carefully drawn using high resolution bit-mapped graphics, and the smooth movements of the runner and the doves are a result of the Commodore's excellent sprite facilities. The music, while not the best example of what can be achieved using the Commodore 64's SID (sound interface device) chip, makes good use of the three available oscillators. The general impression is that the program exploits the capabilities of the 64 to the full, however.

Players can choose to compete in all of the events, one of the events, practise an event or see a list of the world records in each event. There is also an option to allow one or two joysticks to be used, which lets the players compete directly against each other in the swimming and running events without the need for a computer pacemaker.

The first of the eight events is the pole vault, which is perhaps the most difficult of all. The players compete in turn, responding to a series of prompts from the computer. After the lowest bar height of four metres is set, the player is asked whether he wishes to compete at that height. If the answer is yes, the computer then asks for the position of the pole grip, and the athlete then begins the run across the screen.

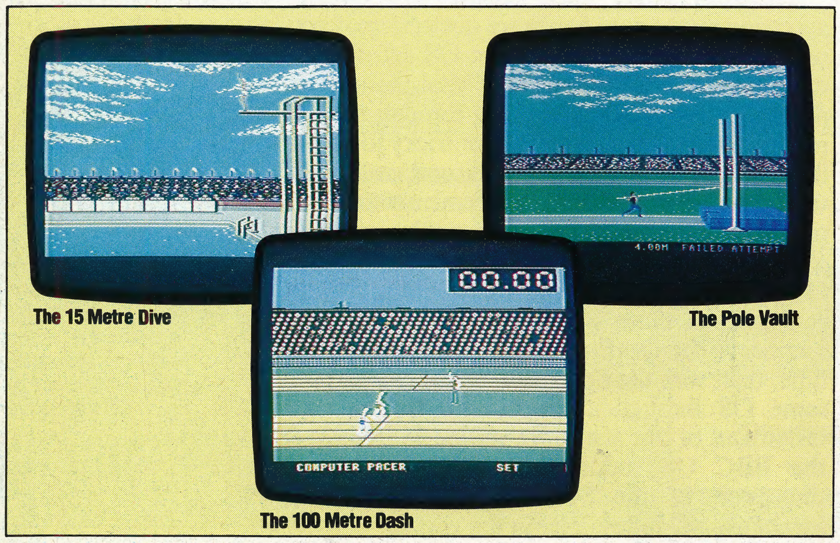
The player must pull the joystick back to plant the pole, push it forward to raise the athlete over the bar and then press the fire button to let go of the pole and prevent it from toppling over onto the bar. Each of these actions requires split-second timing, as a miscalculation at any point will result in the bar being brought down.

This sequence is a good illustration of the kind of parameters that have to be considered while coding. Not only does the screen have to be fully supported at all times to ensure smooth scrolling of the graphics, but the computer also has to check for movements in the joystick and fire button. Finally, the computer has to check that the pole and athlete are at the right angle and position for a successful jump. Throughout this, the user must not notice that these actions are being performed.

After each event, a table is drawn up showing the medals awarded and the winning country's national anthem is played. The computer then loads the next event. The fact that each event is loaded separately shows how much code is required to run each event.

The events that follow are the high diving and gymnastics. Here, the player must manipulate the joystick to produce a smooth dive or vault, making sure to finish with a smooth entry into the water, or land carefully on the mat. The computer then evaluates the performance.

Up The Pole
The quality of Summer Games from Epyx is clearly seen in these photographs. Each event is loaded individually from disk or tape, which allows large amounts of hi-res data for the backgrounds to be stored and processed



The 15 Metre Dive

The Pole Vault

The 100 Metre Dash

IAN McKINNELL

Summer Games: For the Commodore 64. Disk version £19.95; cassette version £14.95
Publishers: Epyx Software, 1043 Kiel Court, Sunnyvale, California, USA
Authors: Randy Glover, Stephen Landrum, John Leupp, Brian McGhie, Stephen Mundry, Erin Murphy, Scott Nelson
Joysticks: Required
Format: Twin cassette or disk versions

DATABASE

Here, courtesy of Oric Products International, we publish the 6502 programmers' reference card

MNEMONIC	DESCRIPTION	R1	R2	IMPLIED	IMMED	ZERO PAGE	Z-PAGE X	Z-PAGE Y	INDIRECT X	INDIRECT Y	RELATIVE ADDRESS	ABSOLUTE ADDRESS	ABSOLUTE X	ABSOLUTE Y	INDIRECT	P-REGISTER							
																N	V	*	B	D	I	Z	C
JSR	JUMP TO SUBROUTINE	PC/S										20											
LDA	LOAD ACCUMULATOR	A			A9	A5	B5		A1	B1		AD	BD	B9		×						×	
LDX	LOAD REG-X	X			A2	A6		B6				AE		BE		×						×	
LDY	LOAD REG-Y	Y			A0	A4	B4					AC	BC			×						×	
LSR	LOGICAL SHIFT RIGHT	A/M		4A(A)		46	56					4E	5E			0						×	×
NOP	NO OPERATION			EA																			
ORA	INCLUSIVE OR ACCUMULATOR	A			09	05	15		01	11		0D	1D	19		×						×	
PHA	PUSH A ONTO STACK	S	A	48																			
PHP	PUSH P-REGISTER ONTO STACK	S	P	08																			
PLA	PULL ACCUMULATOR FROM STACK	A		68												×						×	
PLP	PULL P-REGISTER FROM STACK	P		28												×	×	×	×	×	×	×	×
ROL	ROTATE LEFT ONE BIT	A/M		2A(A)		26	36					2E	3E			×						×	×
ROR	ROTATE RIGHT ONE BIT	A/M		6A(A)		66	76					6E	7E			×						×	×
RTI	RETURN FROM INTERRUPT	P/ PC/S		40												×	×	×	×	×	×	×	×
RTS	RETURN FROM SUBROUTINE	PC/S		60																			
SBC	SUBTRACT WITH CARRY	A			E9	E5	F5		E1	F1		ED	FD	F9		×	×					×	×
SEC	SET CARRY	P		38																			1
SED	SET DECIMAL	P		F8																1			
SEI	SET INTERRUPT MASK (DISABLE)	P		78																	1		
STA	STORE ACCUMULATOR	M	A			85	95		81	91		8D	9D	99									
STX	STORE REG-X	M	X			86		96				8E											
STY	STORE REG-Y	M	Y			84	94					8C											
TAX	TRANSFER A TO X	X	A	AA												×						×	
TAY	TRANSFER A TO Y	Y	A	A8												×						×	
TSX	TRANSFER S TO X	X	S	BA												×						×	
TXA	TRANSFER X TO A	A	X	8A												×						×	
TXS	TRANSFER X TO S	S	X	9A																			
TYA	TRANSFER Y TO A	A	Y	98												×						×	